

Industrial Image Processing

Christian Demant · Bernd Streicher-Abel
Carsten Garnica

Industrial Image Processing

Visual Quality Control in Manufacturing

Second Revised Edition

Dipl.-Ing. Christian Demant
Dipl.-Ing. Bernd Streicher-Abel
Dipl.-Ing. (FH) Carsten Garnica
NeuroCheck GmbH
Stuttgart
Germany
www.neurocheck.com book@neurocheck.com

Authors of the first edition, 1999: Demant, Streicher-Abel, Waszkewitz

ISBN 978-3-662-49591-9 ISBN 978-3-642-33905-9 (eBook)
DOI 10.1007/978-3-642-33905-9
Springer Heidelberg New York Dordrecht London

<http://extras.springer.com/2013/978-3-642-33904-2>

© Springer-Verlag Berlin Heidelberg 1999, 2013
Softcover reprint of the hardcover 2nd edition 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

As a student of ETH Zurich, I encountered image processing for the first time in the mid-1980s. Then, the subject was primarily discussed from a scientific and theoretical point of view (algorithms) and had no practical use in automation technology. Expensive special hardware with weak, non-standardized processors in combination with error-prone assembler programming resulted in poor reliability for industrial processes and thus prevented its spread.

While writing my doctoral thesis in the early 1990s as one of the first Ph.D. candidates at the Paul Scherrer Institute in Zurich (now the Centre Suisse d'Electronique et de Microtechnique, abbreviated CSEM), I focused on this subject with research concerning the then novel CMOS image sensors. Since then, image processing has become the central focus of my professional career.

In the wake of the rapid development of PC technology, the triumphant progress of industrial image processing began in the mid-1990s and continues to this date. Modern industrial production processes are inconceivable without image processing systems. Many automation solutions are even made possible only by using image processing. Industrial image processing has turned from an abstract science into a still ambitious, yet also extremely useful key technology of modern automation technology.

The authors deserve credit for providing the first edition of this book back in 1999, a reference book on the subject of industrial image processing for the first time offering both beginners and advanced readers an ideal introduction and reference. This is not an abstract work of academia but explains in an understandable way the methodical processes and mathematical foundations of important image processing functions. It also deals with all vital aspects needed to implement industrial image processing systems for quality control in industrial manufacturing processes. From illumination to optics, cameras and image capturing hardware, the fundamental software algorithms and automation interfaces, the relationships of all relevant parts are presented.

What makes this book unique is the practical relevance. Using the professional image processing software NeuroCheck developed by authors Demant and Streicher-Abel, the reader is able to follow the many examples in the book taken from practice using an intuitive, modern graphical interface, and parameterize

them anew interactively. From the viewpoint of my former academic work at the institute this is a revolutionary approach.

Therefore, for many interested in image processing the book has rightly become a standard reference within a short time after its publication. And it is still an authority even if the image processing user of today usually does not need to develop algorithms since standard software is available on the market enabling him to implement even complex applications. The understanding of the interaction of all components described in the book is still vital and valid.

Since the publication of the first edition in 1999, many things have changed in the area of image processing hardware, e.g. imaging sensors. The availability of modern digital cameras with ever faster CCD and CMOS sensors, and of modern digital interfaces such as USB, IEEE 1394 (“FireWire”) and Gigabit Ethernet have contributed to image processing becoming even faster and more productive. Modern multi-core CPU technology allows for more comfortable and more reliable image processing software—while simultaneously cost is decreasing.

The authors allow for this development in this heavily revised second edition, and not least thanks to the new NeuroCheck software version 6.0 (available since 2009), they demonstrate what state-of-the-art image processing systems can look like. This standard reference in its latest edition should have its place on every bookshelf.

Frauenfeld, Switzerland, January 2013

Dr. Oliver Vietze
CEO & Chairman
Baumer Group
www.baumer.com

Preface

Since the publication of the first edition of our book in 1999, machine vision has enjoyed continuous strong growth as in the decade before. After machine vision had crossed the 1 billion Deutsche Mark revenue line in Germany in 2000, the same euro milestone was then reached in 2005. The average growth rate was approximately 6.4 % between 2000 and 2010 [VDMA (German Engineering Association)]. One has to look carefully to find industries with comparable growth dynamics.

However, this glossy image has experienced its first setbacks. In 2009 in the wake of the global economic crisis, companies in this industry suffered significant losses (−21 %) for the first time.

In addition to this, machine vision had by now reached the phase of a “consolidated industry” in the life cycle of an economic sector. The spirit of optimism from the 1990s has mostly evaporated, technological quantum leaps have become rare, and by now the continuous reduction of system cost is at a premium. Start-ups can only establish themselves on the market with the help of huge grants and only rarely do they leave the “small business” sector. On the other hand, the number of co-operations is increasing and many market players are growing solely by purposefully acquiring smaller businesses.

Where in the 1990s a complex algorithm was able to convince on the spot, today software reliability during continuous production and trouble-free integration into networked production structures are vital.

Since all industry partners feel the increasing time pressure, intelligent easy-to-use functionality becomes more and more important. Wherever possible, system providers have to use high performing hardware and software standards since the development of proprietary systems is no longer acceptable to the market, neither technologically nor financially.

However, the subject retains its fascination and there is a number of reasons why, globally, machine vision will continue to grow successfully over the following years.

Ensuring quality is the top priority among manufacturers. Machines that are able to “see” gauge high-precision parts, guide robot arms into the correct position, and identify components during production flow from incoming to outgoing goods.

Let us summarize: today, industrial production without machine vision is unthinkable! Therefore, visual inspection systems can be found in businesses of all sizes and industrial sectors.

Especially German industry with its strong “Mittelstand” (medium-sized businesses) again and again holds numerous very different and demanding tasks for machine vision. Hence, German machine vision businesses are globally leading in many areas, especially when it comes to versatility, flexibility and integration into various production environments. Excellent competence with regard to solving the image processing task is the fundamental requirement to be seriously taken into consideration as a provider. With this background, a practical introduction into image processing is now more needed than ever before.

This book is based on years of practical experience on the part of the authors in development and integration of automated visual inspection systems into manufacturing industry. We have tried to use a different approach than most books about (digital) image processing. Instead of introducing isolated methods in a mathematically systematic sequence, we present applications taken with few exceptions from industrial practice. These image processing problems then motivate the presentation of the applied algorithms, which focuses less on theoretical considerations than on the practical applicability of algorithms and how to make them work together in a consistently designed system. The mathematical foundations will not be neglected, of course, but they will also not be the main focus of attention.

We hope that this approach will give students and practitioners alike an impression of the capabilities of digital image processing for the purposes of industrial quality control. We also hope that it will create an understanding for the prerequisites and methodology of its application.

We would like to thank Baumer Optronic, Radeberg, Germany, for the many years of successful cooperation and constructive support in writing the chapter on digital cameras.

We would also like to thank Industrial Vision Systems Ltd., Kingston Bagpuize, UK, for providing and editing the vivid application examples in the chapter “Color Image Processing”.

Furthermore, we want to thank all the people who have supported us in the past years and have been, in one way or another, involved in the evolution of NeuroCheck. With their work and effort, our NeuroCheck brand has become a resounding success and has thus enabled us to produce this book. We wish to thank:

- Dipl.-Inf. Marcellus Buchheit, Edmonds/Seattle, U.S.A.
- Dipl.-Ing. (FH) Richard Herga, Süßen, Germany
- Bernd Marquardt, Dormagen, Germany
- Prof. Dr. Konrad Sandau, Darmstadt, Germany
- Dipl.-Ing. (FH) Anton Schmidt, Bernau, Germany
- Dipl.-Ing. (FH) Axel Springhoff, Metzingen, Germany
- European Patent Attorney Dipl.-Ing. Christoph Sturm, Wiesbaden, Germany

- Dr.-Ing. Peter Waszkewitz, Kornwestheim, Germany
- Earl Yardley, B.Eng. (Hons), Wantage, UK

Finally we would like to express our special thanks to Ms. Hestermann-Beyerle and Ms. Kollmar-Thoni, Springer Publishing, for enabling us to publish this second edition, and, last but in no way least, to our translator, Ms. Michaela Strick.

Stuttgart, spring of 2013

Dipl.-Ing. Christian Demant
Dipl.-Ing. Bernd Streicher-Abel
Dipl.-Ing. (FH) Carsten Garnica
NeuroCheck GmbH
book@neurocheck.com
www.neurocheck.com

Contents

1	Introduction	1
1.1	Why Write Another Book About Image Processing?	1
1.2	Possibilities and Limitations	3
1.3	Types of Inspection Tasks	5
1.4	Structure of Image Processing Systems	6
1.4.1	Hardware	6
1.4.2	Signal Flow in Process Environment	9
1.4.3	Signal Flow Within the Image Processing System	11
1.5	Process Model	13
1.6	Introductory Example	15
1.6.1	Optical Character Recognition	16
1.6.2	Thread Depth	18
1.6.3	Presence Verification	20
1.7	From Here	21
	References	24
2	Overview: Image Preprocessing	25
2.1	Gray Scale Transformation	26
2.1.1	Look-Up Tables	26
2.1.2	Linear Gray Level Scaling	28
2.1.3	Contrast Enhancement	29
2.1.4	Histogram Equalization	30
2.1.5	Local Contrast Enhancement	31
2.2	Image Arithmetic	33
2.2.1	Image Addition and Averaging	34
2.2.2	Image Subtraction	35
2.2.3	Minimum and Maximum of Two Images	37
2.2.4	Shading Correction	37
2.3	Linear Filters	38
2.3.1	Local Operations and Neighborhoods	39
2.3.2	Principle of Linear Filters	40
2.3.3	Smoothing Filters	43
2.3.4	Edge Filters	48

2.4	Median Filter	52
2.5	Morphological Filters	54
2.6	Other Non-linear Filters	57
2.7	Coordinate Transformations	58
2.8	Integral Transformations	60
2.9	Summary	61
	References	63
3	Positioning	65
3.1	Position of an Individual Object	65
3.1.1	Positioning Using the Entire Object	66
3.1.2	Positioning Using an Edge	68
3.2	Orientation of an Individual Object	70
3.2.1	Orientation Computation Using Principal Axis	71
3.2.2	Distance-Versus-Angle Signature	73
3.3	Robot Positioning	75
3.3.1	Application	76
3.3.2	Image Processing Components	76
3.3.3	Position Determination on One Object	78
3.3.4	Orientation of an Object Group	78
3.3.5	Comments Concerning Position Adjustment	79
3.4	Summary	81
	References	82
4	Overview: Segmentation	83
4.1	Regions of Interest (ROIs)	83
4.2	Binary Segmentation	84
4.2.1	Thresholds	85
4.2.2	Threshold Determination from Histogram Analyses	86
4.2.3	Gray Level Histograms	87
4.2.4	Generalizations of Thresholding	90
4.3	Contour Tracing	92
4.3.1	Connectedness	92
4.3.2	Generating Object Contours	93
4.3.3	Contour Representation	95
4.4	Template Matching	95
4.4.1	Basic Operation	96
4.4.2	Optimizing Template Matching	99
4.4.3	Comments on Template Matching	103
4.4.4	Edge-Based Object Search	104
4.5	Edge Detection	106
4.5.1	Edge Probing in Industrial Image Scenes	106
4.5.2	Edge Search with Subpixel Accuracy	108

4.6	Summary	109
	References	111
5	Mark Identification	113
5.1	Bar Code Identification	113
5.1.1	Principle of Gray-Level-Based Bar Code Identification	114
5.1.2	Types of Bar Codes	115
5.1.3	Examples for Industrial Bar Code Identification	117
5.1.4	Two-Dimensional Codes	119
5.2	Character Recognition	120
5.2.1	Laser-Etched Characters on an IC	121
5.2.2	Basic Configuration of Character Recognition	123
5.2.3	Fundamental Structure of a Classifier Application	126
5.2.4	Position Adjustment on the IC	130
5.2.5	Improving Character Quality	135
5.2.6	Optimization in Operation	137
5.3	Recognition of Pin-Marked Digits on Metal	138
5.3.1	Illumination	138
5.3.2	Preprocessing	139
5.3.3	Segmentation and Classification	140
5.4	Block Codes on Rolls of Film	142
5.5	Print Quality Inspection	144
5.5.1	Methods	145
5.5.2	Print Quality Inspection in Individual Regions	146
5.5.3	Print Quality Inspection with Automatic Subdivision	148
5.6	Summary	149
	References	150
6	Overview: Classification	151
6.1	What is Classification?	151
6.2	Classification as Function Approximation	153
6.2.1	Basic Terms	153
6.2.2	Statistical Foundations	155
6.2.3	Defining Classifiers	156
6.3	Instance-Based Classifiers	158
6.3.1	Nearest Neighbor Classifier	158
6.3.2	RCE Networks	159
6.3.3	Vector Quantization	161
6.3.4	Template Matching	161
6.3.5	Comments on Instance-Based Classifiers	162
6.4	Function-Based Classifiers	162
6.4.1	Polynomial Classifier	163

6.4.2	Multilayer Perceptron-Type Neural Networks.	164
6.5	Comments on the Application of Neural Networks.	167
6.5.1	Composition of the Training Set.	167
6.5.2	Feature Scaling.	167
6.5.3	Rejection.	168
6.5.4	Differentiation from Other Classifiers.	169
6.6	Summary.	170
	References.	172
7	Gauging.	173
7.1	Gauging Tasks.	174
7.2	Simple Gauging.	174
7.2.1	Centroid Distances.	175
7.2.2	Contour Distances.	178
7.2.3	Angle Measurements.	182
7.3	Shape Checking on a Punched Part.	183
7.3.1	Inspection Task.	183
7.3.2	Modeling Contours by Lines.	184
7.3.3	Measuring the Contour Angle.	186
7.4	Angle Gauging on Toothed Belt.	186
7.4.1	Illumination Setup.	187
7.4.2	Edge Creation.	189
7.5	Shape Checking on Injection-Molded Part.	191
7.5.1	Computing Radii.	191
7.5.2	Comments on Model Circle Computation.	194
7.6	High Accuracy Gauging on Thread Flange.	194
7.6.1	Illumination and Image Capture.	195
7.6.2	Subpixel-Accurate Gauging of the Thread Depth.	196
7.7	Calibration.	196
7.7.1	Calibration Mode.	197
7.7.2	Inspection-Related Calibration.	199
7.8	Summary.	199
	Reference.	201
8	Overview: Image Acquisition and Illumination.	203
8.1	Solid-State Sensors.	203
8.1.1	Introduction.	204
8.1.2	CCD Sensors.	206
8.1.3	CMOS Sensors.	208
8.1.4	Special Types.	210
8.1.5	Color Sensors.	211
8.1.6	Properties of Sensors.	213
8.2	Digital Cameras.	217
8.2.1	Control of Image Capture.	217

8.2.2	Capturing Color Images.	219
8.2.3	Characteristic Values of Digital Cameras.	221
8.2.4	Operating Conditions in Industrial Environments	222
8.3	Image Data Transfer	223
8.3.1	CameraLink®	224
8.3.2	FireWire®	226
8.3.3	USB	228
8.3.4	Gigabit Ethernet	229
8.4	Line-Scan Cameras	231
8.4.1	Types of Line-Scan Camera Applications	232
8.4.2	Spatial Resolution of Line-Scan Cameras	234
8.4.3	Illumination for Line-Scan Cameras	234
8.4.4	Control of Line-Scan Cameras	235
8.5	Optical Foundations	236
8.5.1	f-number	236
8.5.2	Laws of Imaging	238
8.5.3	Depth of Field	242
8.5.4	Typical Capturing Situations	246
8.5.5	Aberrations	248
8.5.6	Lens Determination.	249
8.5.7	Special Lens Types.	251
8.6	Illumination Technology	253
8.6.1	Light Sources.	253
8.6.2	Front Lighting	255
8.6.3	Back Lighting	258
8.7	Summary.	261
	References	264
9	Presence Verification	265
9.1	Presence Verification Using PTZ Cameras	266
9.1.1	Inspection Part Geometry	266
9.1.2	Illumination	267
9.1.3	Positioning.	268
9.1.4	Object Detection.	268
9.1.5	Verification of Results	269
9.2	Simple Gauging for Assembly Verification	270
9.2.1	Illumination	272
9.2.2	Inspection Criteria	273
9.2.3	Object Creation and Measurement Computation.	275
9.2.4	Position Adjustment	276
9.3	Presence Verification Using Classifiers	277
9.3.1	Illumination	279
9.3.2	Check of Crimping	281
9.3.3	Type Verification of the Flange	286

9.4	Contrast-Free Presence Verification	290
9.5	Presence Verification Using Line-Scan Cameras	292
9.5.1	Inspection of Cylindrical Parts with Area-Scan Cameras	292
9.5.2	Inspection of a Valve Body	295
9.5.3	Notes	300
9.6	Summary	301
10	Overview: Object Features	303
10.1	Basic Geometric Features	303
10.1.1	Enclosing Rectangle	303
10.1.2	Area and Perimeter	304
10.1.3	Centroid Coordinates	308
10.1.4	Axes and Radii	309
10.2	Shape Descriptors	310
10.2.1	Contour Curvature	310
10.2.2	Fiber Features	312
10.2.3	Euler Number	313
10.2.4	Moments and Fourier Descriptors	314
10.3	Gray Level Features	315
10.3.1	First-Order Statistics	315
10.3.2	Textural Features	315
10.4	Summary	316
	References	318
11	Color Image Processing	319
11.1	Color Identification	320
11.1.1	Evaluation Strategy	321
11.1.2	Illumination and Image Capture	322
11.1.3	Color Classification	324
11.1.4	Selecting a Camera Image for Character Recognition	327
11.1.5	Recognition of Writing	329
11.2	Color Segmentation	332
11.2.1	Illumination	332
11.2.2	Color Classification	333
11.2.3	Segmentation	336
11.2.4	Presence Verification	336
11.3	Summary	337
	References	338
12	Implementation of Industrial Image Processing Applications	339
12.1	Image Processing Projects	339
12.2	Process Integration	341

12.3 Outlook	344
Reference	345
Appendix A: Mathematical Notes.	347
Appendix B: Software Download	359
Appendix C: Weblinks to Industrial Image Processing	361
Index	363

Chapter 1

Introduction

With ever increasing demands regarding product quality and documentation, industrial vision has become a key technology. Meanwhile the use of industrial vision systems in automated manufacturing goes without saying. However, there is in many cases a lack of understanding for this modern technology. This book was written in order to remedy this condition, which was in part created by the vision industry itself. As with all areas in which PCs are increasingly used, a trend to give the user more possibilities for application development became apparent in image processing. This makes it also necessary to equip the user with adequate know-how.

In this introductory chapter we will present the typical application areas for vision systems in industry and their basic structure, describe the object-oriented model on which our method is based, and illustrate this model using a simple example. But before doing this, we want to explain why we thought it necessary to add another book on industrial vision to those that are already available.

1.1 Why Write Another Book About Image Processing?

There are a number of books available on digital image processing. It is therefore justified to ask: why add another one? From our experience, the books available can be divided into three categories:

- Most books introduce methods and algorithms, one after the other, in a more or less mathematical fashion. These books are mainly written by (and for) academics and document the ongoing research in the field of image processing. As such they are of immeasurable value to the developers of image processing software. To the end-user, however, who needs to solve a specific visual inspection task, they are of no great help. He starts out with a description of his problem rather than with isolated methods of whose existence he, as a non-expert, may not even know. Furthermore, the methods are usually discussed independently, whereas a solution for an inspection problem will always require the collaboration of several algorithms—which may sometimes yield surprising results.

- Some books deal with the practical development and implementation of image processing software, usually in the form of algorithm libraries. Again, very important for the software developer, they are probably even less useful for the end-user who should not have to concern himself with implementation details of the software he uses for solving his problem.
- A small number of books present real-world industrial applications, which is just what the industrial user needs. Most of the time, though, these books describe only very superficially how the experts arrived at the final solution. The reason for this is that the manufacturers of inspection systems base their competitive advantage on hiding the solution approach in a black box, offering only an extremely limited interface to the end-user. The end-user will typically not be able to get detailed information about the structure and inner workings of the application he bought.

In contrast to this, we are convinced that industrial image processing will only be able to meet expectations if it emerges from its present state as some kind of “occult science” only mastered by a select few and becomes a generally recognized and familiar technology. This book was written to further such a development by describing functioning solutions of real-world inspection problems to show how the various well-known algorithms can actually be used in such a way that they support and enhance each other. Our approach assumes a certain scenario of the future development in the field of image processing, which we will briefly describe in the following paragraphs.

Generally recognized and observed standards are a *sine qua non* for the widespread distribution of a technology. The most important tool for industrial vision is the computer, and the most commonly used standard in this area is a PC with a Windows[®] operating system by Microsoft (Redmond, WA, U.S.A.). Of course there will always be tasks that exceed the limits of a PC system, but the better part of industrial vision tasks can be solved by a PC. The fact that PCs are widespread in private, administrative and industrial areas serves as an additional acceleration factor since most people are familiar with handling mostly standardized user interfaces.

In this we agree with Jähne et al. (1995) regarding the development of image processing systems: “falling prices and the possibility of using familiar computer systems for image processing will make it a tool as generally and easily used as data acquisition is today.”

Image processing software went the same way as software for data acquisition: towards user-friendly, interactive systems, which can be configured and re-tooled by the end-user. This has removed one of the most important obstacles to the application of industrial image processing, especially in small companies. These companies frequently manufacture varying pieces in small series. In this situation, the maintenance cost of an inspection system requiring outside knowledge and an expensive expert to adapt the system to a change in production would be intolerable.

However, improvements in the handling of inspection systems must not obscure the fact that industrial image processing is not and will not be a simple field. Too many factors influence the results: the interactions of test piece, environment and software are too complex. As always in engineering, nothing can replace experience. The expert will still be needed, especially for the initial design and installation of an inspection system. We hope that this book will be a first step for practitioners and students to become vision experts themselves. A second goal of this book is to give an overview of digital image processing enabling decision-makers to understand the technical problems and the process of implementing a visual inspection system even if they do not intend to get so deeply involved with details as would be necessary to design their own vision systems.

Digital image processing is a vast field of work. Examples are the best way of learning in such an area and therefore constitute the core of this book, motivating both the theoretical explanations and the descriptions of algorithms. You can download all you need in order to carry out these examples on an off-the-shelf PC with current Windows operating systems using exactly the same software system employed for the industrial solutions—this should probably be a unique opportunity. Because of this example-oriented, “intuitive” approach, you will not find the most arcane details of every algorithm in this book. We will of course present the essential methods and their mathematical foundations, but our aim is to illustrate the use, application, and effect of the algorithms, not to prove their mathematical validity.

To illustrate our intentions with a handy example: this book does not try to answer the question “What is a hammer, how do I make one and how do I pound in a nail with it?” but encourages the reader to ask himself/herself: “I have a box with a hammer, nails and other tools, how do I use this to build a table or perhaps even a log cabin?” Sometimes we will have to jump ahead of the theory and use methods which will only later be described in detail, but we think this is justified by the possibility of using realistic examples.

1.2 Possibilities and Limitations

It is due to its very visual nature, of all things, that industrial vision is sometimes in a less than enviable situation compared to related areas. Most potential users of automated inspection systems are perfectly willing to accept the difficulties of interpreting endless series of measurements. Even for acoustic data—for which humans also have built-in sensory equipment—these mathematical difficulties are usually appreciated. Manufacturers of image processing systems, however, will frequently hear the argument “But I can easily see that!” What is forgotten is that we humans have learned vision through millions of years of evolution. What is easy for us, is anything but for a machine. One of the main problems in the implementation of automated visual inspection systems is therefore understanding

the way in which the machine “sees” and the conditions that have to be created for it to perform its task optimally.

Directly related to this problem is another difficulty encountered when one tries to introduce image processing systems on the production line: they will inevitably be compared to the peak performance of humans. Of course it is true that people can in general recognize characters without errors, even hardly legible handwriting after adequate practice. It is therefore justified to speak of a recognition rate of 100 %. However, no-one can keep up this performance over the course of a full working day. Although printed characters are easier to recognize, it is fair to assume that the error rate for this kind of visual inspection in industry is even higher than for the reading of handwritten texts because of the failing concentration due to the monotony of the work.

One could easily write several books on the capabilities of the human visual system and how it differs from the processing of image information by a computer. This cannot be the task of this practically-oriented introduction to image processing, which is why we will restrict ourselves to a core statement: automated visual inspection systems are able to deliver excellent recognition results continuously and reliably, equal to the average performance of humans over time, even better in some areas, provided the following basic rules are observed:

- The inspection task has been described precisely and in detail, in a way appropriate for the special characteristics of machine “vision”.
- All permissible variants of test pieces (with regard to shape, color, surface etc.) and all types of errors have been taken into account.
- The environmental conditions (illumination, image capturing, mechanics etc.) have been designed in such a way that the objects or defects to be recognized stand out in an automatically identifiable way.
- These environmental conditions are kept stable.

There must be no doubt that an automatic visual inspection system *like any other machine* has specifications outside of which one cannot expect the machine to function without fault. It is surprising how often this simple rule is ignored for primarily software-based systems. No-one would use a drilling machine equipped with a wood bit to work his way through reinforced concrete, but a program is expected to deal with input data practically unrelated to its original task. Of course, one of the reasons for this is that the users of image processing systems typically do not take the trouble to specify the tasks of the system and the possible variations of the pieces to be inspected in necessary detail and with appropriate precision—although on these issues there are specific and far-reaching requirements concerning the cooperation between the customer who orders such a system and the contractor.

1.3 Types of Inspection Tasks

You can categorize inspection tasks for image processing systems according to the intended goal or the process structure.

Categorization according to intended goal: Steger et al. (2008) subdivide the tasks for image processing systems in industrial manufacturing into the following categories:

- Object recognition
- Positioning
- Completeness check
- Shape and dimension check
- Surface inspection

We basically agree with this categorization. It should be noted, however, that object recognition is a component of many applications without being the actual objective of the respective inspection task. Therefore, we have changed the above categorization, focusing on the basic technology used for marking an object expressly for identification purposes. We will complement this list with two areas that have come into focus over the past years due to rapid technological progress: color image processing and 3D image processing. These are not different tasks rather a different kind of information whose evaluation and capture necessitate special methods. We have also added the category *image and object comparison* because certain types of completeness checks are easier to describe in this way. This leads to the following categorization:

- Positioning
- Mark identification
- Shape and dimensions check, gauging
- Completeness check
- Color processing
- Image and object comparison
- Surface inspection
- 3D image processing

The application areas are listed above in the sequence in which they will be discussed in this book. We will start with position recognition because this type of application has a quite simple structure: as soon as the object has been found, only a single step is left to be done: the position determination. In contrast, we discuss the completeness check towards the end of the book because, notwithstanding the simple name, it can be a very complex application in practice.

In the interest of a coherent presentation and to avoid going beyond the scope of this volume, we will restrict ourselves to the first five application areas which PC-based vision systems are typically used for. A special case of image comparison, print quality inspection, will be briefly discussed in connection with identification. We will glance at surface inspection in the chapter on presence verification. Over the

past years, 3D image processing has been the object of much attention; however, this area is still characterized by a variety of capturing techniques, each with its specific advantages and disadvantages, one of which we will discuss as an example in the overview chapter on image capturing and illumination. Usually 3D image data is evaluated using the methods of classical two-dimensional image processing substituting brightness information with distance information thus the evaluation strategies presented in this book can also be used for 3D images.

Between the chapters on application areas we have inserted overview chapters that discuss certain aspects from the preceding application chapter in greater detail. The overview chapters thus serve to explain the algorithms which are often simply taken for granted in the application chapters.

1.4 Structure of Image Processing Systems

This section gives a short overview of the fundamental setup of image processing systems in industrial manufacturing. This overview is only intended as a first introduction and will therefore not go into details like lighting equipment, properties of cameras or communication with higher-level production control systems. These aspects will be covered more comprehensively in Chaps. 8 and 12.

1.4.1 *Hardware*

Practically every image processing system can be roughly divided into three parts: sensors, computer, and communication interfaces, as depicted in Fig. 1.1. One area has been omitted, although it is often the decisive factor for the success of image processing applications: lighting, which is too difficult to generalize for a self-contained description of the system setup. We will try to make up for this in Chap. 8.

Sensors: The sensors of a system for visual quality control are typically cameras, as shown in Fig. 1.1, although other image-producing sensors can also be used, e.g. laser and ultrasonic sensors. Scanners of the kind used in graphics design and for the analysis of photographic material, e.g., satellite images, are rarely used in industrial applications, above all because of their slowness. Camera technology is discussed in detail in Chap. 8.

The connection between sensors (i.e. cameras) and computer is usually achieved via digital media such as FireWire, Gigabit-Ethernet or USB. These PC mass market technologies have established themselves in industrial applications over the past years thus proving to be the logical extension of an effect typical of the PC sector: profits from the mass market are used to drive the development in the industrial high tech sector.

Computer: Depending on the application, very different types of computers may be used. Parallel computers are often used for the extremely data-intensive

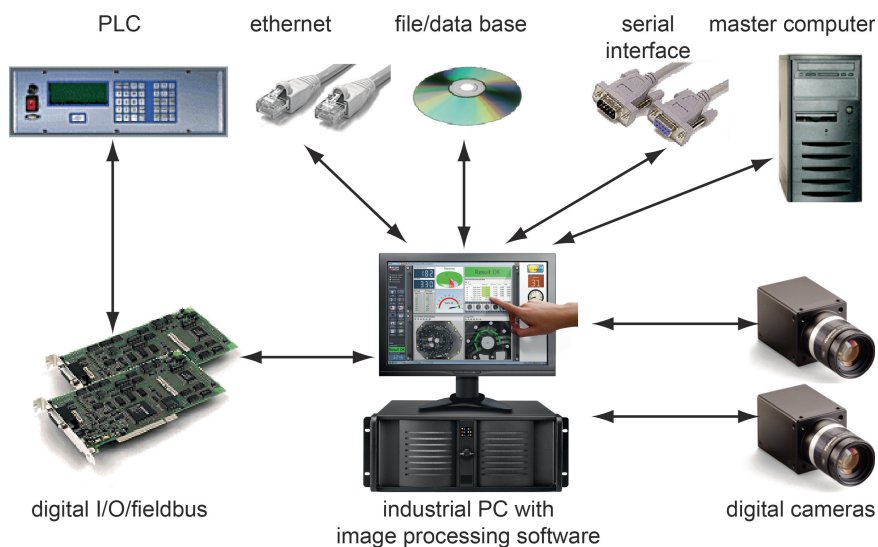


Fig. 1.1 Industrial vision system

inspections of continuous manufacturing processes like steel, paper or textile production, because workstations or PC systems do not provide sufficient memory bandwidth and computation speed to handle the data rates of such applications. The bulk of industrial inspection tasks can easily be handled with PCs and standard components, though. By using modern multi-core CPUs, industrial vision profits immensely from a quantum leap in PC system performance. Especially the time-consuming computation of image data can be distributed over the various processor cores—proper, intelligent multi-threaded implementation provided—thus frequently leading to significantly shorter evaluation times.

Until the mid-1990s, PC systems were not a serious competitor in industrial image processing, mainly because of insufficient bandwidth of their bus systems. VME bus systems and specialized vision processors dominated the market. No other segment of information technology has developed as rapidly over the past decades as the PC sector. The increase in performance with a simultaneous decrease in prices allows for the solving of demanding image processing tasks with the help of PCs. This is a kind of positive feedback, a self-accelerating effect: the widespread use of the PC architecture makes expensive hardware and software development worthwhile, which opens up new performance and application ranges; this in turn increases the attractiveness of the PC platform, and so on. Also the high level of standardization with regard to hardware and software interfaces contributed to the fact that PC systems today play an important role in all areas of industry, from manufacturing control to quality inspection. Another example is the frequent use of PC technology in the area of programmable logic control (PLC) systems.

Communication: An image processing system for industrial quality control has to work in step with the manufacturing process, i.e. it must be possible to control the system from the outside. The system, on the other hand, must be able to transmit its results to an external control in such a way that they can be processed in automated production and quality control systems. The image processing system must therefore be capable of communicating with other devices.

For remote control and immediate evaluation of final results (test passed or failed), image processing systems are often connected to programmable logic controls using digital interfaces or a Fieldbus. The system can also be connected directly to a master computer using a network or serial communication. All these means of communication can coexist. Usually the PLC is directly responsible for the synchronization of inspection system and production process whereas the master computer is responsible for global control and logging of quality data. Of course, the image processing system itself can record quality-relevant data, like measurements and the like, in files for further processing. By using standardized file formats, this data can be evaluated practically everywhere—another advantage stemming from the widespread use of PC systems. Taking this idea a step further, we come to the concept of remote maintenance of inspection systems, e.g., over the Internet. This part of customer support is of great importance when supporting image processing systems, enabling suppliers to support their clients over large distances within minutes. Despite these obvious advantages, visual inspection lines without external network access are still installed because of security concerns thus consciously forgoing the option of remote control maintenance. Because of the decisive economic advantages, it can be presumed that in the medium term most systems will be equipped with a remote control maintenance option.

Intelligent cameras: Beginning in the middle of the last decade (circa 2005) there was a trend towards the development and use of intelligent cameras. By the start of the current decade this market seems in decline, a point that we would like to comment on. In principle, intelligent cameras follow the hardware setup outlined above, but the computer is integrated into the camera casing. The advantages of this type of system are the small size and low cost of purchase. This lets them appear attractive as a first step into the world of image processing, in particular for small and medium-sized companies. On the other hand, computation performance and especially the memory capacity of these cameras are limited due to their small size, so that they are only suitable for relatively simple applications. Depending on the camera type, the application has to be programmed, usually in C, or has only a very limited set of user-adjustable parameters. In effect, this is a miniaturization of the old “black box” concept to withhold information from the user. Building powerful, object-oriented inspection applications in this way is very difficult. Also, these systems can often visualize the inspection process and results only in a limited way.

As an added advantage, simplified operation in comparison with a PC system is often mentioned. This has to be taken with a grain of salt since the configuration of inspection applications can usually not be carried out directly on the camera, but often requires an additional PC as a terminal. The inspection application will then

be configured on the PC—by programming it or by setting parameters of predefined routines available on the processor of the camera—and downloaded to the camera, usually by Ethernet or serial interface. Consequently, frequent re-configuration and optimization—as is typical for the initial operation of a production process, but also occurring later due to changes in the product spectrum or simply because of drifting production parameters—are rather tedious.

This is not to deny the usefulness of intelligent cameras. One should be very clear, however, on the present capabilities of such systems and their limitations compared to those of PCs that we have grown accustomed to. The calculation of the economic efficiency of intelligent cameras must usually be repeated when using two or three cameras. On the other hand, a PC equipped with the proper software can evaluate the images of a dozen cameras or more without difficulty.

1.4.2 Signal Flow in Process Environment

The purpose of an industrial image processing system is to derive a quality statement from an image scene, i.e. something that exists in the real world. Simplified as far as possible, the signal flow of an image processing system can be represented by Fig. 1.2. Figure 1.2 shows that an image processing system is connected to the outside world via at least two interfaces. Of course, further interfaces are possible for remote or manual control of the system, but the two interfaces illustrated above are indispensable: on the input side of the system the real-world scene is translated into an image to be processed by the computer; on the output side the processing result is transferred to the environment as a quality statement.

Output interface: The quality statement can be made in very different ways. This holds for content as well as for signal technology. It can be a numerical value, a good/bad statement, a string of characters or even something totally different; it can be transferred over a data line, printed, stored in a file or displayed as a light signal. All this depends entirely on the task itself and on the process environment. In any case, some kind of symbolic representation within the image processing system has to precede transfer of a statement to the outside world.

Input interface: As we have already mentioned in the previous paragraph, very different types of sensors can be used to provide the image information on the input side. Basically, the result is always the same: a digital image encoding the brightness of the image scene as numerical values (this also applies to color images, but then each of the base colors red, green and blue will require its own brightness value). A digital image forms a matrix of brightness values. Of course,

Fig. 1.2 Schematic signal flow of an image processing system

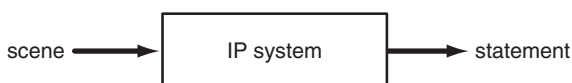
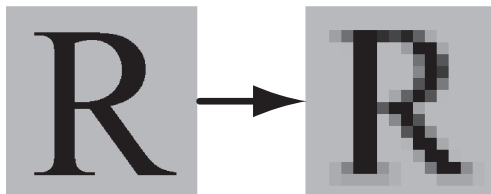


Fig. 1.3 Effects of sampling in digital image processing system



this image is only true to reality to a certain extent. Two factors are especially important here: sampling and quantization.

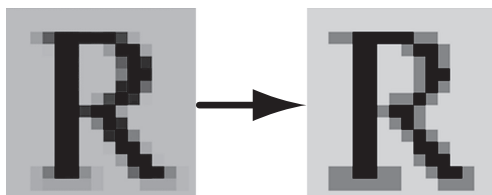
Sampling. Digital images have only a limited number of pixels.¹ Through the process of digitization, the real image scene is fitted into a limited grid of pixels. Chap. 8 will describe the characteristics of this process in more detail, mathematically as well as technologically. For the time being it may suffice to say that the inevitable loss of information follows Shannon's sampling theorem, which has a simple interpretation here: The sampling interval, i.e. the distance between two pixels in the digital image must not exceed half the size of the smallest relevant detail in the scene (Sonka et al. 2008). This observation is especially important for gauging tasks, because it limits the theoretical measurement precision.

Figure 1.3 illustrates the sampling effect using the conversion of a character into a digital image of limited resolution. Note that the original image contains only two brightness levels, a medium gray as background and black for the character. The resulting image, however, contains intermediate gray levels caused by the distribution of light on the different sensor cells. This allows the recovery of part of the information lost in the sampling process, which will be described in more detail in Chap. 7.

Quantization. A point in a scene can have any brightness. Within certain limits given by the background noise and the saturation level of the sensor this is also true inside the camera (if we assume for the moment that a CCD or CMOS camera is used as it usually is in industrial image processing today). Inside the computer, though, the brightness of each pixel is represented as a numerical value and the value range of a computer is inevitably limited. For various reasons, e.g. memory limits, computation time, easy handling or simply because the task does not require more effort, digital image processing software often uses a much smaller range of values than theoretically possible. Nowadays, gray level cameras offer resolutions of ten or twelve bit, i.e. 1,024 or 4,096 brightness levels; however, in industrial vision a resolution of eight bit is usually used, even today. The original brightness signal is usually quantized into 256 levels. Accordingly, color cameras can provide ten or twelve bit brightness resolution for each of the three primary colors; however, in industrial vision only eight bit, i.e. 256 levels per primary color are usually used. For special tasks such as spectroscopy or distance images, images

¹ Actually this is true for every image, even those seen by the human eye or taken as a photograph, but in these cases the resolution is much higher so that the sampling is not recognized.

Fig. 1.4 Effects of quantization in digital image processing system



with significantly better gray level resolution are used, however, this requires more technical effort. Another reason why using 256 gray levels is so popular is that this resolution perfectly fits the typical memory organization of today's computers into bytes of 8 bits which can represent exactly 256 different values.

Reducing a potentially unlimited number of possible brightness values to 256 gray levels sounds much more dramatic than it is. Actually, the human eye itself does not reach better quantization. According to Russ (2007) it is hardly possible for the human visual system to distinguish more than 30 gray levels on a video monitor or a photograph. Typical output devices, like laser or ink-jet printers, are subject to similar limitations. Figure 1.4 shows the effect of gray level quantization on the sampled character from Fig. 1.3—strongly exaggerated so that the effect is visible despite the limitations of the printed reproduction.

Figure 1.5 illustrates the importance of the information loss through sampling and quantization in an image processing application using the image of a spark plug used in the introductory example in Sect. 1.6. Left to right you see the original image, the result of a sampling operation and that of a quantization of the sampled image. After sampling, the smaller characters on the spark plug become illegible and the thread appears blurred. After quantization the object can no longer be distinguished from the background in some places. The combination of sufficient spatial and brightness resolution is decisive for the solution of a visual inspection problem. Information lost at this point can only be recovered to a very limited degree.

1.4.3 Signal Flow Within the Image Processing System

As explained in Sect. 1.4.2, processing in a vision system starts with the sampled and quantized image transmitted by the sensor and ends with a symbolic statement to be transmitted to process peripherals. Using the recognition of a single character as an example, this can be depicted as follows:

- It starts with the sampled and quantized image of the character. This is also called the *iconic*² level because there is no additional information available on the content of the image.

² Greek (eikon): image.

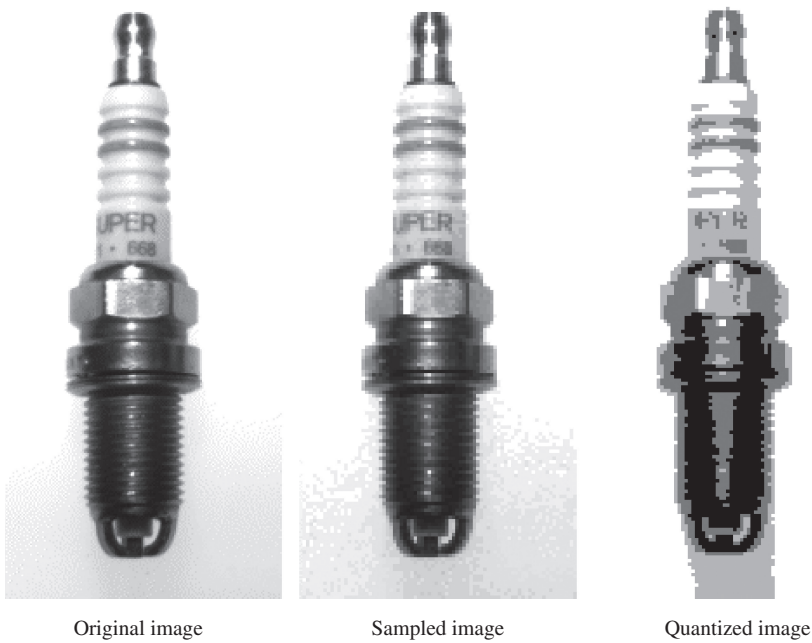


Fig. 1.5 Effect of sampling and quantization on an image

- Since the computer is not able to recognize “at a glance” that the image shows a certain character, the image undergoes a series of processing steps. At the end of this process, the character is represented by an *object*—as it is understood in computer science: the unstructured representation of the character as a matrix of brightness values has been transformed into a data structure describing an object found in the image scene using properties like position, size, shape etc. This process of isolating identifiable objects from the originally unstructured image area is called *segmentation*.
- Finally, the information to be transmitted to the environment is derived from the properties of the objects segmented within the image scene. In this case, the information is the name of the character, in other applications it could be a dimension or the number of objects. In any case it will be information that can be represented by symbols understandable to humans. Therefore, this stage is sometimes called the *symbolic* level of image processing.

Figure 1.6 illustrates the above sequence of processing steps. The transition from the purely iconic level to the description of a segmented object is symbolized by the depiction of the isolated character in uniform black without background, whereas the extraction of symbolic information is indicated by using an actual character from a computer font.

Fig. 1.6 Signal flow within an image processing system

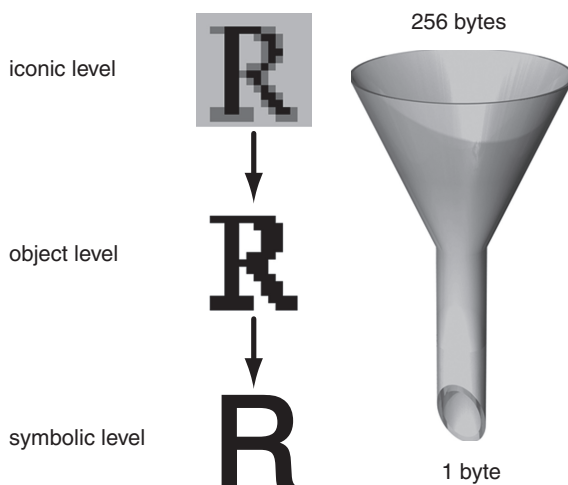


Figure 1.6 also shows that a considerable reduction of information takes place during processing. At the beginning we have an image of a character consisting of 256 pixels in a $16 * 16$ grid. Using the common representation of each pixel by one byte (with 256 possible brightness values) this amounts to 256 bytes of information. At the end of the chain we have the symbolic information showing that this image represents the character 'R'. In a typical computer font this information is represented in a single byte. The amount of information has been reduced by a factor of 256, while *knowledge* has been gained simultaneously. After the complete processing sequence has been carried out and the image processing system has delivered its result, we know that the image contains a certain character. This is a very typical phenomenon in image processing: a large amount of unstructured information contained in an image is restructured and condensed step by step into a tightly confined piece of symbolic information.

1.5 Process Model

There are various approaches to design the sequence of algorithms for solving an image processing task. Which approach to use depends on the task itself and the image material: Using digital image processing for analyzing satellite images is very different from industrial quality control with regard to the properties of the images, their typical content, and the objectives. The approach is also affected by the tools used for solving the problem. To a certain degree, every image processing system will influence the way the user works—unless he goes to the trouble of programming every single function and the whole application frame on his own. And, finally, it is a matter of personal taste, as can be seen from the fact that different authors favor different approaches.

We do not have space to discuss all the various approaches here, nor is this the intention of this book. Of course, we, too, favor a specific approach which, based on our experience, we consider to be appropriate for the majority of industrial applications and which is therefore reflected in the software we have developed. Since all examples presented in the following have been solved using this software and are therefore based on our particular solution pattern, we think it appropriate to outline this approach as a guideline before we proceed with the examples.

The typical aim of an industrial visual inspection is to check the compliance of a test piece with certain requirements, e.g. regarding prescribed dimensions, serial numbers, presence of components etc. The complete task can frequently be subdivided into several independent parts, each checking a specific criterion. These individual checks typically run according to the following model:

1. Image capture
2. Image preprocessing
3. Definition of one or more (manual) regions of interest
4. Segmentation of objects
5. Computation of object features
6. Decision as to the correctness of the segmented objects.

Capturing an image, possibly several for moving processes, is of course a prerequisite for analyzing a scene. In many cases these images are not suited for immediate examination and require preprocessing to change certain properties of the image, e.g. enhancing contrast, suppressing noise, emphasizing specific structures etc. You will get to know a number of these preprocessing methods in the various examples of the following chapters. They will be discussed in detail in Chap. 2.

In most cases it is at least approximately known which image areas have to be analyzed, e.g. where a mark to be read or a component to be verified is located. Therefore, we set Areas of Interest (AOI) or Regions of Interest (ROI). Of course, such a region can also comprise the entire image. Restricting image analysis to these areas serves to accelerate processing and to increase flexibility by allowing different areas to be treated differently. It is possible to be looking for completely different information in the various areas of an image whose characteristics are so different they cannot be processed with the same method. Thus we could be looking for light writing in one area and dark writing in another at the same time since processing parameters can be set for each area individually.

Industrial image processing typically requires the determination of properties of certain objects present in the image scene. The next step therefore is to isolate these objects, a process called *segmentation*. Because of the essential role of this processing step, various segmentation methods are discussed in detail in Chap. 4. After the objects have been segmented, characteristic properties can be computed, such as area, perimeter, position, orientation, distance from each other, similarity to predefined patterns (e.g., for character recognition). Finally, these properties are checked for compliance with the nominal values of the inspection task.

It should be mentioned that there are sometimes applications not requiring segmentation. An example is surface inspection, where one can often simply evaluate certain features of an image area (like the brightness distribution). In this case the characteristic properties can be computed directly from a manually defined region of interest. For this reason, our approach does not distinguish between manually defined regions of interest and segmented objects. Naturally, these two types of structures exhibit different behavior in certain respects and not every operation is meaningful for both types (e.g., features like position and size are always predefined for a manual region of interest and do not need to be computed), but in general they can be considered equivalent. We would like to stress this point in particular, because it is an unusual approach but allows for a high flexibility, as you will see later in our examples. On the one hand, in our approach, all features available for actual image objects can also be computed for manually placed regions of interest. On the other hand, every object segmented from the scene can be immediately used as a region of interest, restricting subsequent operations to this area. This allows for a direct representation of the hierarchical structures prevalent in technology—like screws inside a casing, components on a circuit board—in the structure of the image processing system.

1.6 Introductory Example

The example presented in this section will familiarize you with the fundamental concepts of industrial image processing. In the design of vision systems, many different aspects have to be considered simultaneously, corresponding to the parallel nature of visual information processing. The sequential structure of a book is not ideally suited for such a subject and will force us in the more complicated examples of the following chapters to use algorithms which are only later discussed in detail. Therefore, we will give a brief overview of various fields and methods in this introductory chapter so that we can later refer to these concepts.

The introductory example demonstrates three of the most frequent and important application areas of digital image processing:

- Character recognition as a special case of the application of pattern recognition methods in identification technology
- Completeness check
- Gauging

We will not go into the details of the algorithms used here; instead, we will focus on the introduction of terms and on the approach to a visual inspection task.

Inspection task: A spark plug is to be inspected for the following criteria:

1. Three-digit type number on the ceramic body
2. Thread depth
3. Presence of two washers.

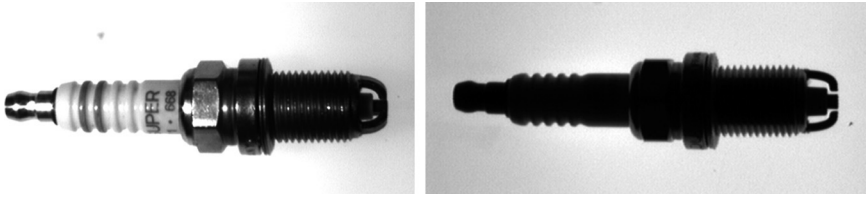


Fig. 1.7 Spark plug with front and back lighting

Figure 1.7 shows the spark plug under two different illuminations: the front lighting required for character recognition, and the back lighting which is advantageous for gauging.

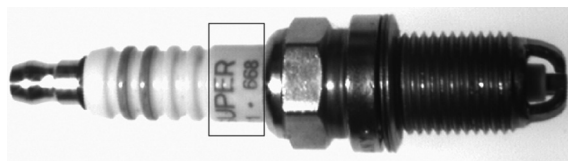
Program hint: You'll find this example in the directory `Examples\Introduction\` of the download.

1.6.1 Optical Character Recognition

The recognition of the type number is the most clearly structured subtask: first the characters have to be found in the image; then they will be presented to a classifier capable of assigning the correct name for the depicted character; the result will then be compared to the expected type number. According to our approach from Sect. 1.5 the solution is as follows:

1. *Image capture.* The image from the camera is digitized and transferred to the computer's main memory. The result in our example is Fig. 1.7.
2. *Define ROIs.* For the recognition of the type number, the area to be searched for characters can be limited to a small section of the ceramic body. Limiting the search area has two advantages: the time needed for the search decreases with the size of the region and certainty is increased: other image areas might also contain characters or similar structures which would make a correct assignment more difficult. Figure 1.8 shows the search area used here.
3. *Segmentation of objects.* Inside the defined search area each character is now isolated as a separate object. This procedure is called *segmentation*. We will neglect for the time being how the segmentation is actually performed; overview Chap. 4 will discuss segmentation methods in detail.
4. *Computation of object features.* For a human being, each of the objects is immediately recognizable as a digit; the machine however has to derive from a more or less arbitrary collection of light and dark pixels which digit is present.

Fig. 1.8 ROIs for searching type number



This means that the machine has to identify the individual segmented objects. Chapter 6 will provide more detailed information on *classifiers*. For the moment it will suffice to regard a classifier as a black box that receives certain input signals and outputs a class designation. In this example certain properties of the individual digit objects are fed into the classifier, which then computes the class of the object from these properties, i.e. which digit it represents.

For printed characters, the image of the character itself is often used directly as input to the classifier, though certain additional features can considerably ease the task of the classifier. But even if the image alone is used as the input signal, some preprocessing will be required. Classifiers typically have a fixed number of inputs. Since the image of the character contains many individual feature values—one brightness value for each pixel—it has to be assured that differences in the size of the characters are compensated. In this step, we will therefore carry out such a normalization of the characters. Figure 1.9 shows the segmented objects, indicated by a frame, and the corresponding input images for the classifier (the images have been rotated into the normal orientation for printing; for the classifier this orientation is unimportant).

5. *Decision.* In the last step the segmented objects are classified using the identically-sized resampled images from the previous step. The correspondence of the recognized classes to the expected values is then checked. Figure 1.10 shows the result of this classification.

You will certainly have noticed that one of the steps from Sect. 1.5 is missing from the above procedure: image preprocessing. When possible, one tries to do without preprocessing because preprocessing algorithms usually have a simple structure and process the entire image, so that they require a lot of computation time. Most importantly, preprocessing should never be used as a substitute for carefully optimized lighting and optics because information lost in the sensor chain is virtually unrecoverable, at least not with acceptable effort. This fact is frequently underestimated by those with no or little experience in image processing.

The distinction between preprocessing and image analysis—which starts in our approach with the creation of objects—is fluid in many cases. For example, most

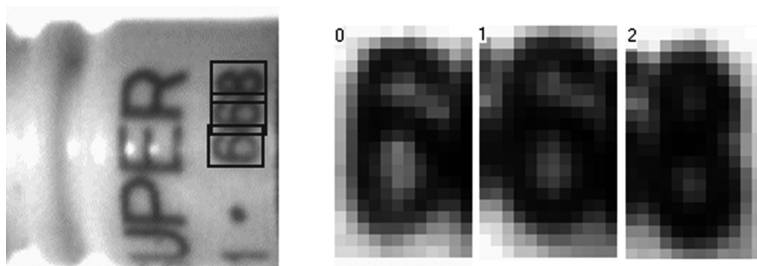
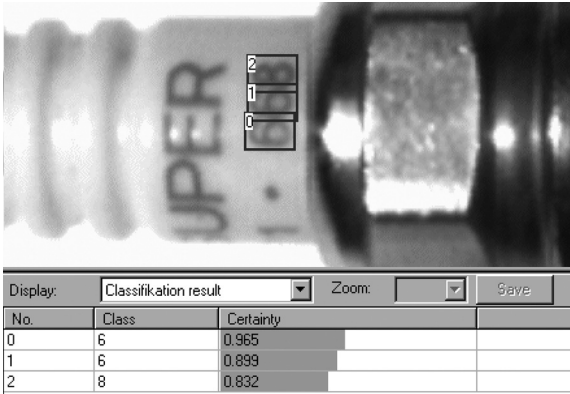


Fig. 1.9 Segmented and size-normalized digits

Fig. 1.10 Result of classification



books on image processing first create a binary image, i.e. an image containing only black and white, to achieve segmentation, assuming that one of the colors represents the complete background, the other all existing objects. By definition, this is a preprocessing operation since a new image is created that is similar to the source image. By contrast, you can create the objects directly from the source image without generating a real binary image in between thus omitting one preprocessing step.

Some details are of course missing from the above procedure which will be discussed later in connection with the various application areas. Let us now turn to the second subtask.

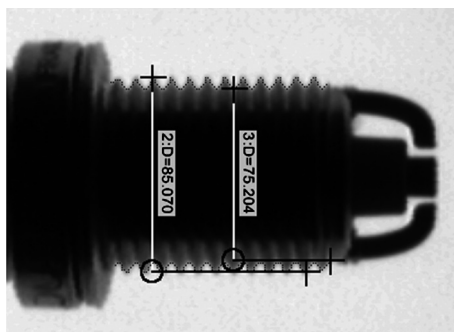
1.6.2 Thread Depth

Two different types of errors have to be distinguished for the inspection of the thread depth. The thread might be cut too deep or not deep enough, or not at all. These two types of errors have to be described in such a way that they can be determined visually, because you cannot order a computer simply to “check whether the thread has the right depth!”

Let us start with a thread cut too deep. If we take a cross section of the top of the spark plug and draw two lines parallel to its axis so that they just touch the tips of the deepest turns of the thread on each side, the distance of these lines can be used to determine whether the thread is too deep: if the lines are too close together, i.e. the distance is too small, then the thread is cut too deep.

In the second case, when the thread is not cut deeply enough (or not at all), the difference between the largest outer diameter and the inner diameter of the thread will be too small. Just as in the first case we can take a cross section of the top of the spark plug and draw two lines parallel to its axis. This time, though, the lines will not touch the innermost but the outermost tips of the thread. The inside diameter is already known from the previous check so that the difference can now be computed easily. Since outside diameter as well as inside diameter move within narrow

Fig. 1.11 Principle of thread depth gauging



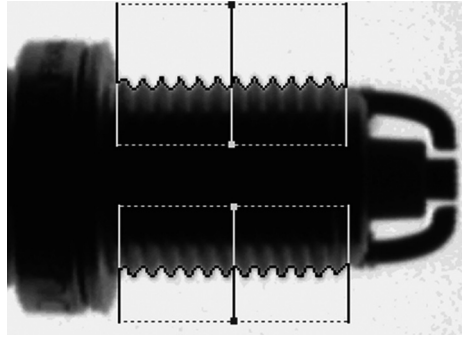
tolerances, it will not even be necessary to actually compute the difference. Instead, each measurement can be checked separately. Figure 1.11 illustrates the principle.

The inspection runs basically as before, but the segmentation stage differs considerably from that of the character recognition subtask. Of course it would in principle be possible to create the entire thread area as an object and determine its minimum and maximum width, but it will be much simpler and faster to trace the outer contour of the object from both sides.

1. *Image capture.* Again the image is captured and transferred to PC memory. Even if the camera resolution were sufficient to carry out all subtasks in a single image, we would still need the second image capture here, because gauging is typically done with back lighting, whereas the character recognition requires front lighting.
2. *Define ROIs.* Following the gauging procedure outlined above, two search lines are defined crossing the thread contour from top and bottom respectively. The contour can be easily found by searching along these lines for a transition from bright to dark. As soon as this change takes place, a point of the contour has been found.
3. *Create objects.* As soon as the first contour point has been found along the search lines, the two contours can be easily created by an algorithm which traces the contour in such a way that there are always light pixels on one side, dark pixels on the other (the particulars of describing and tracing such a contour are explained in Chap. 4). A boundary on both sides of the search line prevents the tracing algorithm from running around the entire spark plug. Figure 1.12 shows the search lines with their boundaries and the generated contour lines.
4. *Compute object properties.* In this step, the minimum and maximum distances of the detected contour lines are computed perpendicular to the axis of the spark plug. For this, parallels are drawn to this axis (purely arithmetically, of course) and shifted until they just touch the outermost, or innermost, pixels of the contours. Since the spark plug is in perfectly horizontal orientation, we can simply use the difference of the y-coordinates to measure the width.³

³ Chapters 3 and 7 discuss how to deal with position shifts and how to perform gauging with an accuracy better than one image pixel.

Fig. 1.12 Search lines and detected thread contours



5. *Decision.* This step is very simple in this case. The computed distances are compared to the nominal values, i.e. it is determined whether the distances lie within prescribed tolerances around the nominal values.

1.6.3 Presence Verification

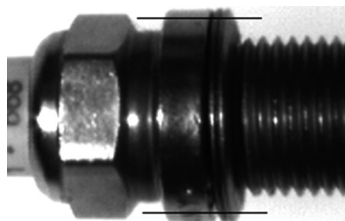
Visual inspection tasks of the *presence verification* type are often as simple to describe as they are difficult to solve. In this case, the aim is to check the presence of the two washers visible in Fig. 1.7 on the base below the thread. The difficulty lies in the fact that it is practically impossible to separate these washers as individual objects from the thread base. Being capable of completing interrupted edges even under unfavorable lighting conditions with ease, humans can clearly see the division between the washers and the thread base. An image processing system cannot achieve this without making extensive use of a priori knowledge about the structure of the scene. Fortunately, there is a much simpler solution which capitalizes on the seemingly troublesome fact that thread base and washers appear as a single object. If the washers are missing, the thickness of this object changes, i.e. its horizontal dimension changes. Using this measurement, the presence of the washers can be easily checked.

It is a very typical method for solving presence or completeness verification tasks: instead of detecting and counting all required objects separately, properties are checked which differ significantly from good to faulty test pieces. Since the segmentation of objects is often the most difficult part of an image-processing application, higher reliability can be achieved using methods which do not try to model exactly the human way of solving such problems.

The solution of the inspection task can then be carried out as follows:

1. *Image capture.* As in the previous subtasks, an image is captured and transferred to PC memory.
2. *Define ROIs.* In order to measure the thickness of the thread base (including the washers), two search lines are used crossing the thread base from right to left.

Fig. 1.13 Thread base with search lines



Figuratively speaking, the spark plug is drilled through on each side with a gauging line as Fig. 1.13 shows.

It is very important to note here that this method will work only if the spark plug always appears in exactly the same position in front of the camera. This cannot usually be assumed in real-world applications which will practically always require position adjustments. Therefore, we will take an immediate look at position adjustment in Chap. 3.

3. *Create objects.* The thread base can be separated from the background in the same way as the thread itself was separated from the background in the previous section. Here, however, we are not interested in the outer contour but in the length of the portion of the search lines running through the thread base. This corresponds to the part of the search lines crossing dark image areas. After the first dark pixel has been found on one of the search lines, the algorithm will not trace the outer contour but will follow the search line until it reaches bright pixels again. The entire dark portion is created as an object. Figure 1.14 shows the two linear objects thus created. They are of course aligned with the original search lines, but exist only within the dark area defined by the thread base and the washers.
4. *Compute object properties.* The length of each of the line objects is computed in this step as the decisive quality criterion for this subtask. Afterwards all objects are removed whose length does not comply with the nominal values. The objects must be neither too short (which would indicate missing washers) nor too long (which would suggest a segmentation problem).
5. *Decision.* In this step, the number of objects is determined that have the correct length. Only if there is exactly one such object on each side will we conclude that segmentation has been successful and that the washers are present. Only then will the test piece be considered “OK”.

1.7 From Here

In the course of this book we will present applications of different types step by step. Except for some illustrative examples, all these applications are actual industrial inspection tasks which have essentially been solved in the way



Fig. 1.14 Result of presence verification using cross sections through thread base and washers

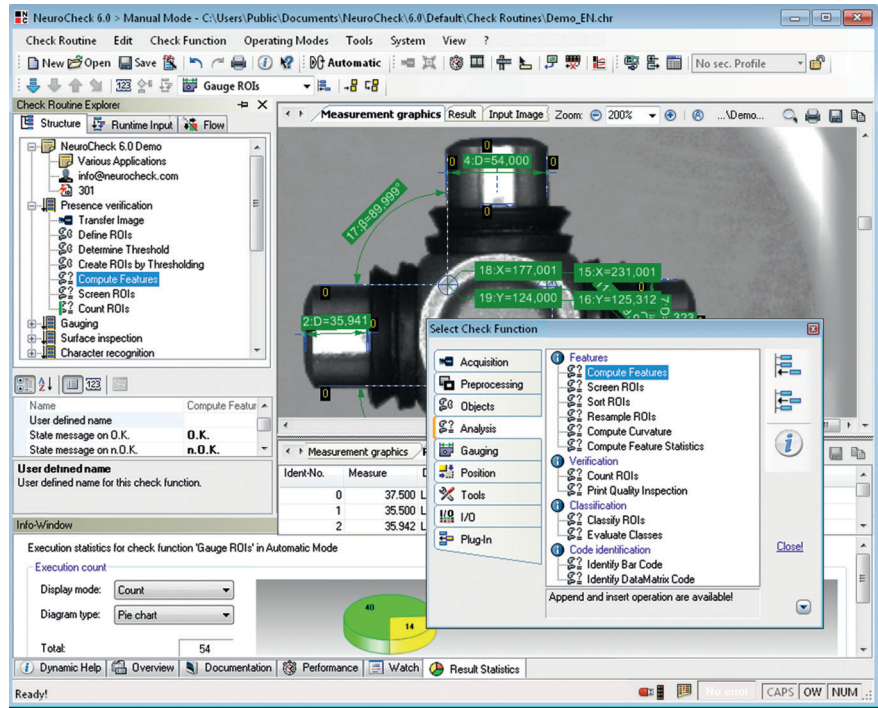


Fig. 1.15 User interface for configuring vision solutions

presented. The qualification “essentially” refers to certain details which we did not find suitable for presentation in an introductory reference book. In a number of cases the companies involved are mentioned. However, this was not always possible mostly for reasons of confidentiality.

Most of the applications use a broad spectrum of algorithms, which was why we have decided to abandon the traditional format of systematically presenting isolated methods ordered according to mathematical considerations. Image processing problems can only be solved with a combination of different methods; presenting the synergistic effects of such combinations was our primary intention. Nevertheless, each application has its particular focus, which is usually supplemented in the subsequent overview chapter.

We have endeavored to keep the chapters as self-contained as possible to enable reading a chapter on gauging at one sitting and obtaining information about classification methods in another reading. It was not possible, though, to avoid interdependencies completely without sinking into endless repetitions. But we have tried to give more answers than cross-references.

In this context we would like to mention that such a book can of course make no claim to be exhaustive. A multitude of methods and algorithms has been developed in recent years in the field of digital image processing, sometimes



Fig. 1.16 Freely configurable visualization for operating staff

adapted to very special requirements. It would by far exceed the scope of an introductory work to go into this kind of detail. Instead, we have concentrated on field-proven methods which can be applied to a broad range of industrial problems. Readers interested in specific algorithms or more comprehensive mathematical background will find a selection of books in the bibliography which go into more detail—one way or another. It is our opinion, though, that it would not befit the character of a text book to amass references to theoretical papers from research journals and that the audience targeted by such a book would not benefit from this. Therefore, we have concentrated on classic works which present a broad selection of methods and—as is appropriate for their more mathematical orientation—an equally wide range of references to the world of image processing research.

Throughout the book you will find paragraphs designated as **Program tips**. This refers to the implementation of the methods or applications presented using the image processing software **NeuroCheck** which is available for download. This software was used to implement the applications presented in this book, i.e. the inspection task solution was interactively configured using the software which then carried out this inspection fully automated in the manufacturing process. The interactive configuration does not only include creating the image processing solutions as described in the following chapters, but also setting up communication with the line's control system and designing visualization for the operating staff. Figure 1.15 presents an impression of the user interface for interactive configuration. An example for visualization for the operating staff is shown in Fig. 1.16. You will find more information about the download in Appendix B.

References

- Jähne B, Massen R, Nickolay B, Scharfenberg H (1995) Technische Bildverarbeitung—Maschinelles Sehen. Springer, Berlin
- Russ JC (2007) The image processing handbook, 5th edn. CRC Press, Boca Raton
- Sonka M, Hlavac V, Roger B (2008) Image processing, analysis, and machine vision, 3rd edn. Cengage Learning, Stamford
- Steger C, Ulrich M, Wiedemann C (2008) Machine vision algorithms and applications. Wiley, VCH-Verlag Weinheim

Chapter 2

Overview: Image Preprocessing

Preprocessing algorithms frequently form the first processing step after capturing the image, as you will see in many of the examples in the following chapters. Therefore, we will start the overview chapters with an introduction to image preprocessing. Gonzalez and Woods (2008) present a comprehensive overview. To give a clear conceptual idea of the effect of the various operations we will use very simple synthetic sample images in this chapter. The application examples in the following chapters use many of these algorithms on real-world industrial images.

Image preprocessing typically denotes a processing step transforming a source image into a new image which is fundamentally similar to the source image, but differs in certain aspects, e.g. improved contrast. An exception to this are the integral transforms which will be briefly discussed in Sect. 2.8. The output image created by such an operation usually exhibits no visual similarity to the source image. Instead it shows certain characteristics of the source image, for example the spectral properties derived by a Fourier transform.

According to this definition preprocessing is changing the brightness of individual pixels or changing their position with respect to each other. The preprocessing functions discussed in this chapter, which change the brightness of individual pixels, can be divided into two basic groups, depending on what the resulting brightness value of a pixel in the output image is derived from:

1. *Pixel operations* use exclusively the brightness of the corresponding pixel in the source image (depending on the preprocessing method and its parameters) to compute the brightness of the output image. This group also encompasses *image arithmetic* functions which combine several images, because these functions only use a single pixel on a fixed position from each of the source images to determine the brightness of the result pixel. Pixel operations can be further divided into *homogeneous* and *inhomogeneous pixel operations*. Homogeneous operations use the same transformation function for each pixel, for inhomogeneous operations the transformation function depends on the location of the pixel in the image. Pixel operations are discussed in the following sections:

- Gray scale transformation (Sect. 2.1)

- Image arithmetic (Sect. 2.2).
2. *Local operations* take a certain neighborhood of the current pixel into account when computing the brightness of the corresponding output image pixel. An example is the mean value filter, which sets the brightness of the output image pixel to the average brightness of a small neighborhood of the corresponding point in the source image. Local operations are discussed in the following sections:
- Linear filter (Sect. 2.3)
 - Median filter (Sect. 2.4)
 - Morphological filter (Sect. 2.5)
 - Other non-linear filters (Sect. 2.6).

The third method introduced in this chapter comprises geometrical transformations that change the pixels' positions with respect to each other (Sect. 2.7).

2.1 Gray Scale Transformation

Following (Sonka et al. 2008), a local gray scale transformation has the form:

$$o_{ij} = f(i_{ij}). \quad (2.1)$$

where o_{ij} is the gray level of the resulting image pixel (*output*) at the coordinates $x = i$, $y = j$ and i_{ij} the gray level at the same position in the source image (*input*). Equation (2.1) therefore means that the same transformation function is used for converting the gray level of every pixel from the source image into the gray level of the corresponding pixel in the output image. The transformation function $f()$ itself does not depend on the position. Otherwise, we speak of *brightness corrections*, for example by relating the image to a reference image (see Sect. 2.2.4).

The transformation function $f()$ is, in principle, arbitrary. For example, images with a very large range of gray levels may suggest a logarithmic transformation. It is not necessary that $f()$ can be expressed analytically. Practical implementations will often use a look-up table to speed up execution, allowing for completely arbitrary transformations (see Sect. 2.1.1).

2.1.1 Look-Up Tables

We will start this section on gray scale transformations with a discussion of look-up tables since they can be used as a tool for all transformations described in the following. They not only allow for transformations which are hard to express analytically, it is also much faster to look up the result of a transformation for one of the typical set of 256 gray levels than actually computing it for each pixel.

In mathematical terms, using a look-up table for a gray level transformation means that the transformation function $f()$ in Eq. (2.1) is given by a table containing one support point of the function for each possible gray level. In effect, this assigns a specific function value to each gray level. Color images can also be transformed using look-up tables where you can either transform all color channels using the same look-up table or, if you want to treat the color channels differently, transform each color channel with its own specific look-up table.

Depending on the complexity of the transformation, considerable speed-up factors can be achieved by using look-up tables, up to a factor of 15 or more for logarithmic or exponential transformations. Figure 2.1 shows examples of the effect of several common look-up tables. The squared look-up table, for example, reduces the overall brightness of the image, but increases contrast by making dark image areas darker, bright ones even brighter. The square root look-up table, on the other hand, increases the overall brightness, but since the effect is less pronounced in bright areas than in dark ones, contrast is reduced. The s-shaped Gaussian or sigmoidal table renders dark and bright areas more homogeneously, respectively, but increases contrast in areas of medium brightness.

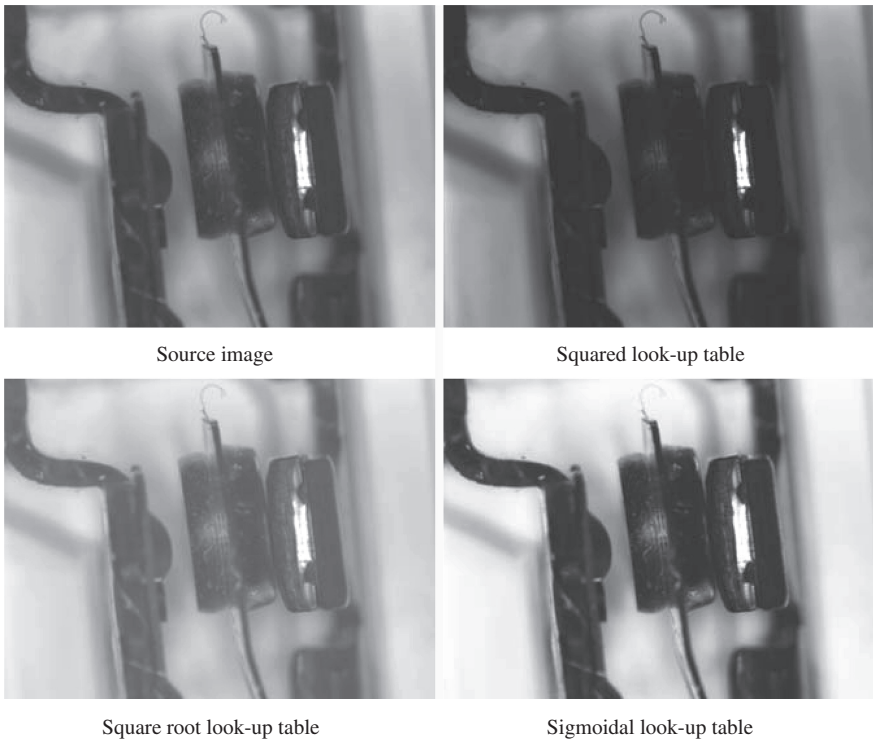


Fig. 2.1 Effect of common look-up tables

In addition to the acceleration achieved by using look-up tables, they also allow for gray level transformations which cannot be written down in a simple way—if at all. Figure 2.2 shows a specially defined look-up table and its effect on the source image from Fig. 2.1. It results in binary thresholding of the source image, i.e. a reduction to only two gray levels. The special shape of the look-up table results in originally very bright and very dark areas appearing black, whereas areas of medium brightness appear white.

Although it is possible to achieve remarkable effects using look-up tables, as shown in Fig. 2.2, a word of warning may be appropriate here. Especially customized look-up tables depend very much on stable lighting conditions and a constant appearance of the test objects in the image. If this cannot be guaranteed, the gray scale transformations have to be adapted dynamically to the current image conditions. This increases complexity considerably, and with it, of course, computation time. Look-up tables of this kind are applicable where satisfactory lighting conditions cannot be achieved, but where the achievable situation can be kept stable.

Program hint: You can process an image with all these look-up tables in **NeuroCheck** by including function *Apply look-up Table to Image* in your check routine.

2.1.2 Linear Gray Level Scaling

An important special case of gray scale transformations according to Eq. (2.1) is linear scaling:

$$o_{ij} = c_2 \cdot (c_1 + i_{ij}). \quad (2.2)$$

where o_{ij} is again the gray level at position $x = i, y = j$ in the output image, i_{ij} the gray level at the same position in the input image. For different values of the coefficients c_1, c_2 the following effects result according to Nischwitz et al. (2007):

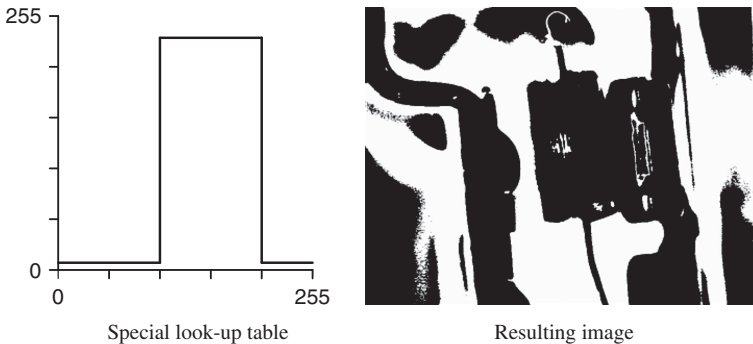


Fig. 2.2 Effect of a specially adapted look-up table

$c_1 > 0$, overall brightness is increased

$c_1 < 0$, overall brightness is reduced

$|c_2| > 1$, contrast is enhanced

$|c_2| < 1$, contrast is diminished.

It is important to clip the gray levels correctly. If the operation were to result in a negative value, gray level 0 (black) has to be used. Values over 255¹ have to be replaced by 255 (white).

2.1.3 Contrast Enhancement

Linear scaling of gray levels according to Eq. (2.2) can be used as a simple but often very effective way of improving image quality by choosing the coefficients appropriately. Starting from the minimum and maximum gray level within the image, one can normalize the contrast of the image so that it uses the full range of gray levels from 0 to 255.

If g_{min} is the minimum, g_{max} the maximum gray level of the source image, contrast normalization is achieved by:

$$o_{ij} = \frac{255}{g_{max} - g_{min}} \cdot (i_{ij} - g_{min}). \quad (2.3)$$

Subtracting g_{min} shifts the minimum gray level of the image to 0, the maximum gray level to $g_{max} - g_{min}$. The subsequent multiplication stretches the gray level range so that the maximum gray level becomes 255. Figure 2.3 shows the effect this transformation function has on an image with very poor contrast.

Strictly speaking this is not a pixel operation in the sense of the above definition. Since g_{min} and g_{max} , the parameters of the operation, are derived from the entire source image, each gray level in the resulting image depends on all gray levels in the source image. Formally, though, the transformation is performed in exactly the same way as an arbitrary scaling with predetermined coefficients. Therefore, we can still consider this a pixel operation from an implementation and practical point of view.

Program hint: In **NeuroCheck** you can improve the overall contrast of an entire image by using function *Enhance Image* and setting the dimensions of the local enhancement areas to the full size of the image.

¹ To avoid even more complicated abbreviations, we will always assume to work with images of 256 gray levels.

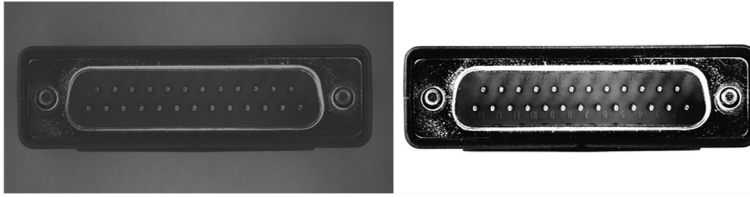


Fig. 2.3 Global gray level enhancement of an image with very poor contrast (enhanced image has been framed to set it off from the background)

2.1.4 Histogram Equalization

In addition to the linear gray scale stretching described in Sect. 2.1.3, several other methods for contrast enhancement exist, for example the histogram equalization. Histograms are tables of the frequency of each gray level in an image. They will be discussed in detail in connection with binary thresholding in Sect. 4.2.3. It is possible to apply an appropriate gray scale transformation to an image so that its histogram corresponds to a given predetermined distribution. In the case of histogram equalization the gray scale transformation shifts the gray values in such a way that each brightness interval comprises an equal number of pixels. This does not necessarily mean that each gray level occurs with equal frequency but rather that each range $[0;g]$ of gray values contains a number of pixels proportional to the width of the interval, i.e. $g/256$.²

Mathematically this works as follows: Every gray level g of the source image is replaced by a gray level g' derived from the relative frequency of all lower gray levels according to

$$g' = 255 \sum_{i=0}^g \frac{N_i}{N}. \quad (2.4)$$

Where N_i is the number of pixels with gray level i in the source image, N the total number of pixels in the source image. The sum is the cumulative frequency of gray levels in the image. This function is monotonic non-decreasing with g which guarantees that the brightness order of two pixels of different gray level will not be reversed. A pixel which was brighter than its neighbor before the transformation will not be darker afterwards.

A comment on implementation: a program will, of course, not calculate Eq. (2.4) for every single pixel. Instead it will first determine the histogram for the complete image, as explained in Sect. 4.2.3. Then it will build a table containing for each gray value g the resulting gray level g' according to Eq. (2.4). The individual image pixels are then converted using this look-up table, just as in Sect. 2.1.1.

² In general, we would have to write this as $g/(g_{max} + 1)$.

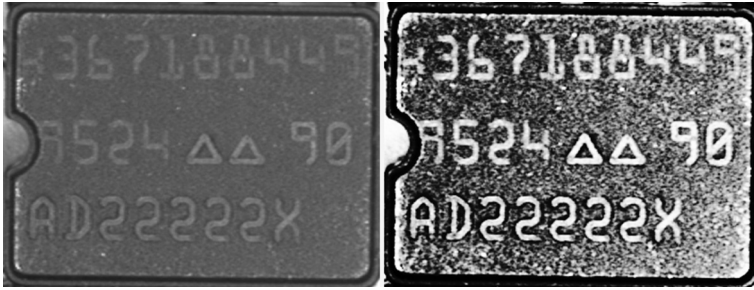


Fig. 2.4 Enhancement of details through histogram equalization

What is the use of this operation? Take, for example, an image containing a few regions of mostly uniform brightness. In such a case, the histogram will exhibit a few peaks whose size corresponds to the areas of these regions. A histogram equalization spreads these peaks out and compresses the areas of less frequently occurring gray levels in-between. The spreading of the peaks has the effect that the differences between gray values within the regions are greater. The regions will therefore be less uniform after the histogram equalization. This makes it easier to recognize faint details as can be seen in Fig. 2.4.

It should be mentioned, however, that methods for contrast enhancement, be it by gray scale stretching or histogram equalization, do not have a fundamental effect on automatic processing of an individual image, because the relation between the brightness of two image pixels is not changed: a brighter pixel remains brighter, only the exact value of the difference changes. The difference between the brightness, however, was already present in the source image. Nevertheless, these methods have their uses. For one, they are very useful if a person has to evaluate the image, in printed form or on a video screen, because the human eye is not able to distinguish shades of gray as delicately as an image processing system. A human's ability to discriminate is thus well-supported by histogram equalization. On the other hand, the reliability of an automated image evaluation can be improved by making sure that the source images are similar in specific characteristics—in this case the brightness distribution. Section 5.2.5 shows an example in which local gray level stretching simplifies object segmentation by reducing contrast differences between source images.

2.1.5 Local Contrast Enhancement

We will end this section on gray scale transformations with local contrast enhancement. Strictly speaking this is not a gray scale transformation in the sense of Eq. (2.1) at all, since the assumption no longer holds that the transformation is independent of the location within the image. Instead, it is only independent of the

location within a specified section of the image. For the following reasons we will describe this important algorithm here:

1. There is no explicit dependence on location in the transformation function $f()$. The coefficients c_1 and c_2 of a linear gray level scaling according to Eq. (2.2) are not written as a function of image coordinates. Instead, a different set of coefficients will be used for different sections of the image. This means that we have a transformation function which is piecewise independent of location.
2. From an implementation and practical point of view, the difference between a transformation equal over the entire source image and one equal for *predefined* rectangular image sections is negligible. The step towards truly local operations where the transformation function changes *for every single pixel* is much larger and more difficult.

Local contrast enhancement differs from the contrast enhancement given by Eq. (2.3) only with respect to the gray level limits g_{max} and g_{min} . These limits will now be determined not for the entire image but individually for each image section. For the sake of simplicity, rectangular image sections of equal size are typically used, but of course this operation can also be restricted to the inside of a segmented image object. Such a local normalization of image contrast can compensate for



Fig. 2.5 Image with poor contrast and illumination gradient in the background, after global and local enhancement (top to bottom)

illumination gradients and remove shadows. Figure 2.5 shows that global contrast enhancement can improve the overall contrast, but it cannot remove the illumination gradient. Local enhancement can improve the contrast in each image section. In contrast to simple look-up tables or gray scale transformations it can also tolerate varying illumination conditions up to a certain degree.

Program hint: In **NeuroCheck**, you can achieve a local contrast enhancement by setting an appropriate field size in check function *Enhance Image* from function group *Preprocessing*.

Any gray scale transformation can be applied locally by subdividing the image prior to the transformation into sections in which the transformation function remains constant, as shown above. If, instead, the sections are moved along with the pixel to be transformed, we have a truly local operation in which the transformation function changes with each individual pixel. According to Jähne (2005) this is called an *inhomogeneous pixel operation*. Such operations usually have to be treated differently from the local normalization described above. This can be illustrated using the optimization by building a look-up table described in Sect. 2.1.4. If the image sections are not predefined but moved together with the pixel, each pixel would require building a new look-up table for its neighborhood. This will only be efficient if the neighborhood is considerably larger than the 256 entries of the look-up table. This will obviously change the characteristics of the algorithm and its computation requirements significantly. In reverse, this justifies including the local contrast enhancement with the pixel operations.

2.2 Image Arithmetic

Image arithmetic denotes the pixel-wise combination of two (or more) images. The gray level of a pixel in the resulting image will of course depend on the gray levels of the corresponding pixels in *all* input images. Since there are no gray levels in a neighborhood of the respective position involved in the computation, this can nevertheless be categorized as a pixel operation. It is an inhomogeneous one, though, because the transformation function can vary for every pixel, depending on the corresponding pixels in the source images.

This can also be seen from Eq. (2.5) which states that the resulting gray value at position (x, y) is computed from the gray values of all input images at the same position according to function $f()$. This function, though, depends on the gray levels of all other input images at the same position, so that it may vary for every single pixel.

$$g_o(x, y) = f(g_0(x, y), g_1(x, y), \dots, g_n(x, y)). \quad (2.5)$$

In the following, we will discuss the typical case of combining only two images. In general, any arithmetical or logical operation of two variables can be

used as the transformation function $f()$. The most important operations in image processing are:

- Addition and averaging
- Subtraction
- Minimum and maximum
- Division for shading correction.

2.2.1 Image Addition and Averaging

Adding two or more images is a simple operation, but displaying and processing them further poses a problem since the resulting values could exceed the gray level range of $[0; 255]$. For the resulting image to be displayed and processed as a gray level image, one of the following methods can be used:

Clipping: If you use this operation, output values higher than 255 are clipped to 255. This method is commonly used in multimedia and therefore part of the instruction set of modern PC processors. It is of little consequence in image processing.

Division by number of images: This method is equivalent to image averaging. It can be used to reduce noise. Each camera adds a certain amount of readout noise to the image. This noise is essentially random in nature unlike the image information; therefore, it will look different in subsequent images of the same scene while the image information itself will remain unchanged (for the most part). A specific pixel will thus be lighter in one image and darker in another. If you average subsequent images, the overall noise will be reduced while the image information remains the same. When dealing with very dark image scenes, it is advisable to integrate the image information in the camera itself using an adequate exposure time and to read it out before a pixel has reached saturation. However, if you are dealing with normally lit scenes, you achieve the best noise reduction according to Russ (2007) by summing up several images captured with normal exposure time in the computer. For effective noise reduction, at least ten images should be used, which takes a long time. This is why such an operation is avoided in industrial image processing through optimized lighting and sensory conditions.

Scaling to minimum and maximum: With this method, the images are first summed up in an output image that is actually capable of storing all possible output values. Then the actual minimum and maximum values have to be determined. This range is scaled to $[0; 255]$; this corresponds to contrast enhancement according to Sect. 2.1.3. Because of the calculation and memory requirements, this method is avoided as well.

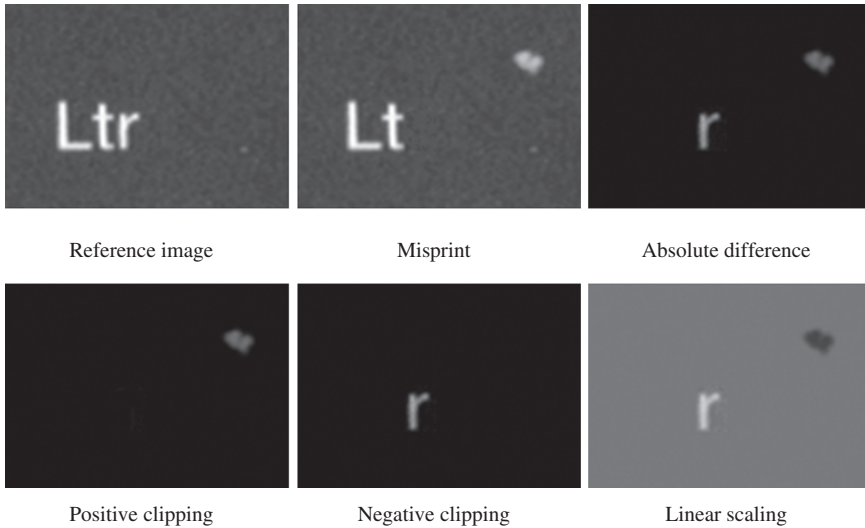


Fig. 2.6 Result of different subtraction scalings for two different types of error in print

2.2.2 Image Subtraction

Image subtraction is more interesting and versatile than image addition. Since negative values can be encountered using this operation, new problems—or new possibilities, depending on your attitude—for result scaling arise. Depending on the desired effect, different scaling methods can be used. Their respective effects are shown in Fig. 2.6.

Absolute value: This scaling method uses the absolute value of the subtraction result. If you are only dealing with two source images, this is absolutely sufficient to limit the results to a range of $[0; 255]$; if you are dealing with more than two images, a division by the number of images minus one becomes necessary (if two images are subtracted from one image, the worst possible result can be -510 ; after you have taken the absolute value and divided by two, the value is back in gray level range). In this case, the resulting image contains information as to how much the source images differ and in which spots. A possible application would be the determination of absence of necessary components without the specification whether these components are dark or light. They just have to differ from the background.

A typical example for applications like this is print quality inspection. What we are interested in here is whether the print is damaged, i.e. whether there are missing sections in the printed characters, and whether there are color spots in the background. Let us suppose white is printed on a dark gray background. A reference image is subtracted from the current camera image. If the print is perfect, the output image is uniformly dark (not completely black since the brightness values of real world objects are never that uniform). If sections of a character are

missing, a large negative value is generated because the camera image shows the gray level value of the background, yet the reference image shows the gray level value of the light print. After taking the absolute value, a bright spot will appear in the resulting image. The same thing occurs if there's an additional color spot in the camera image; then the subtraction will already result in a high gray level value and taking the absolute value does not change this. Both print faults appear as bright spots in the output image after subtraction and can be easily segmented and analyzed afterwards.

To perform this subtraction in a real application involving print quality control is anything but trivial. Minimal variations in position result in major errors since differing areas along the edges of the print are very prominent. We will discuss this example further in Sect. 5.5.

Positive clipping: If you're just interested in finding areas where the first image was brighter than the second, you can use clipping to set all negative result values to 0. In the above example all missing areas within the printed characters would not appear because the subtracted reference image is brighter resulting in a negative result value, which is neglected here. This mapping method is advisable if you only want to determine where and to what extent the first image is brighter than the second, i.e. whether there are additional bright objects or whether required dark objects are missing. The amount of calculation involved is lower than that for the determination of the absolute value.

Negative clipping: You can also do the reverse, set all positive output values to 0 and use the absolute values of the negative ones. In this case, only the absence of bright objects in the first image or the presence of additional dark ones can be detected.

Linear scaling: As with addition, linear scaling to the possible range of gray level values is the most expensive method. For the same reasons as above, the theoretical interval will usually be chosen as a basis for scaling and calculate the mapping of two source images as follows:

$$g_0 = \frac{1}{2} (255 + g_1 - g_2). \quad (2.6)$$

The difference of the source images' gray values is raised by a full gray value interval, i.e. the theoretical output value is changed from $[-255; 255]$ to $[0; 510]$. Then all gray values are divided by two so that the output value is within the $[0; 255]$ range. This guarantees that all pixels for which the gray values of both images are identical receive a medium gray value. All pixels for which image 1 is brighter than image 2 will appear brighter than this medium gray value, all others darker. So the output image not only contains the information where and how much the two source images differ, but also in which direction. This method should be used when these two types of error have to be evaluated differently. This is the case for print quality inspection when a specific maximum number of dark scratches within the print can be tolerated, but light color splashes of the same size cannot.

2.2.3 Minimum and Maximum of Two Images

When combining two images using the minimum operator, each pixel of the output image is set to the gray value of the darker one of the corresponding pixels in the source images. The maximum operator does the same using the lighter one of the two pixels. The minimum operator can be used to determine the number of dark objects in two subsequent images. It prevents objects in identical positions in both images from being counted twice. The maximum operator will be discussed in Sect. 2.3.4, where we will use it to combine edge images. For this, an image is processed using different edge filters, which causes edges of varying alignments to be set off while areas without edges appear very dark. Edges are always the brightest pixels in each output image. The combination of these images using the maximum operator assures that edges of any alignment are set off brightly in the output image, as can be seen in Fig. 2.22.

2.2.4 Shading Correction

Shading correction is a special case of image arithmetic using the otherwise rare operation of image division. What is done here is the correction of systematic brightness deviations. These deviations can have different causes:

1. Border shading: sometimes lenses exhibit a decrease in brightness from the center of the image to the periphery; this is called border shading or vignetting.
2. Uneven lighting, e.g. shadow effects, illumination gradients due to one-sided lighting.

Brightness deviations can be generally represented by:

$$g(x, y) = f(x, y) b(x, y). \quad (2.7)$$

Where $g(x, y)$ is the gray value of the captured image at position (x, y) , and $b(x, y)$ is the true brightness at the same position. $f(x, y)$ is a mapping function that cannot usually be written down or determined as an analytical function. The illumination has some influence, as well as the characteristics of the optical system, and the characteristics of the sensors used.

How can this error be rectified if the error function $f(x, y)$ is unknown? Since everything takes place in a geometrically discrete space of pixels, all you need is the numerical value of $f(x, y)$ at each position. If you had an image scene whose true brightness at each position was known, you could easily calculate the function values of $f(x, y)$ by taking an image of this scene and using it later for correction. This is precisely the idea of shading correction.

First, a reference image is created by capturing an image of an area of uniform brightness under the same conditions that will later on be encountered in operation (strictly speaking the area wouldn't have to be uniform, but it makes calculations

considerably easier). An image is created containing the reference gray values $g_r(x, y)$, from which the correction values can be calculated by $f(x, y) = g_r(x, y)/b$ if the brightness b of the image captured is known. The true brightness $b(x, y)$ of the pixels in the image scene to be inspected can then be calculated from the captured gray values $g(x, y)$ using division by reference gray values and multiplication by reference brightness b , (Nischwitz et al. 2007):

$$b(x, y) = b \cdot \frac{g(x, y)}{g_r(x, y)}. \quad (2.8)$$

This sort of correction is appropriate, according to Russ (2007), for sensors with a linear relation between brightness and voltage; with a logarithmic characteristic, a correction by subtraction is more appropriate and is, in many cases, also sufficient for linear sensors to get an image that can be further processed. Subtractive correction also holds the advantage that it is less computation-intensive. This type of correction is achieved by

$$b(x, y) = b + (g(x, y) - g_r(x, y)). \quad (2.9)$$

The true brightness in an image scene differs from the reference brightness b for each pixel exactly by the difference between the gray value captured in the image scene and that from the reference image. Figure 2.7 shows the result of a shading correction for a label only illuminated from one side. It can be clearly seen how in the original image the last character merges with the background after a binary thresholding because of poor illumination. In the corrected image, thresholding can be done without any problems (cf. Sect. 4.2 for information on binary thresholding). Especially when inspecting print on round objects, e.g. bottles, shading correction has to be applied to compensate for the change in the angle of incidence caused by the curvature.

From a theoretical point of view, the problem remains to determine the true brightness of the reference image. In practice, this usually does not present a problem since one does not have to find out the *true* brightness, but has to provide stable conditions that allow for proper further processing.

2.3 Linear Filters

In this chapter we will present the principle of linear image filters and some commonly used filters and the way they can be employed. We'll do without mathematical analyses of filter design and effect. For further information on the theory please refer to Jähne (2005), Nischwitz et al. (2007) or Sonka et al. (2008).



Fig. 2.7 Image with uneven illumination before and after shading correction

2.3.1 Local Operations and Neighborhoods

Linear filters are some of the most commonly used local image operations. A *local operation* is a preprocessing function in which the gray value of a pixel in the resulting image is determined by a group of pixels in the source image. This group of pixels is called a *neighborhood*, with filter operations it is also called a *mask* or *filter kernel*.

This can be formulated as follows: the gray value g at a position (x^*, y^*) in the resulting image is the result of a mapping $f()$ that is applied to all gray values g_i of all pixels in the source image that are located in the neighborhood N^* (so far not

determined in more detail) of the pixel (x^*, y^*) . Usually the target pixel (x^*, y^*) is also part of this neighborhood, but this is not required. The following formula results:

$$g(x^*, y^*) = f(\{g_i(x, y) | (x, y) \in N^*\}). \quad (2.10)$$

The mapping function $f()$, too, is at first arbitrary. By specifying this function appropriately, completely different types of filters are obtained. When executing such a local operation, the defined neighborhood usually is moved pixel-wise across the image, and this way the output image is calculated pixel for pixel. That is why these methods are ideally suited for parallelization: since the original image is not changed, the calculation can in principle be performed simultaneously for all pixels—if you have sufficient processors at your disposal.

2.3.2 Principle of Linear Filters

Linear filters are obtained by using a weighted sum of the gray values within the defined neighborhood as the mapping function $f()$ in Eq. (2.10):

$$g(x^*, y^*) = \frac{1}{s} \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} f_{cr} g(x^* - dx + c, y^* - dy + r). \quad (2.11)$$

At first glance this formula may look somewhat intricate, but applied to a concrete example it quickly loses its terror and you can recognize what it really means. At first, let us turn to a verbal explanation of the formula, then we will give a quite simple example.

dx and dy are the distances between the target pixel and the upper left corner of the neighborhood, i.e. the filter kernel. Let us assume that the filter kernel consists of R rows and C columns. Correspondingly r is the row index and c the column index of the filter operator. The above equation only states that every pixel within a rectangle of width C and height R , whose upper left corner is located at the position $(x^* - dx, y^* - dy)$ is weighted and summed up with the corresponding filter weight f_{cr} . The result is then scaled with a factor of s . In most cases, s is the sum of all filter weights because in this case the average gray value is preserved, but this is not obligatory.

Figure 2.8 demonstrates the application of this principle in a simple mean value filter. The filter has the frequently used size of 3×3 pixels. All filter weights are set to 1, the target pixel is located in the center. We obtain the following values:

$R = 3$, number of rows of the kernel

$C = 3$, number of columns of the kernel

$dx = 1$, distance between target pixel and left border of kernel

$dy = 1$, distance between target pixel and upper border of kernel

$s = 9$, sum of filter weights.

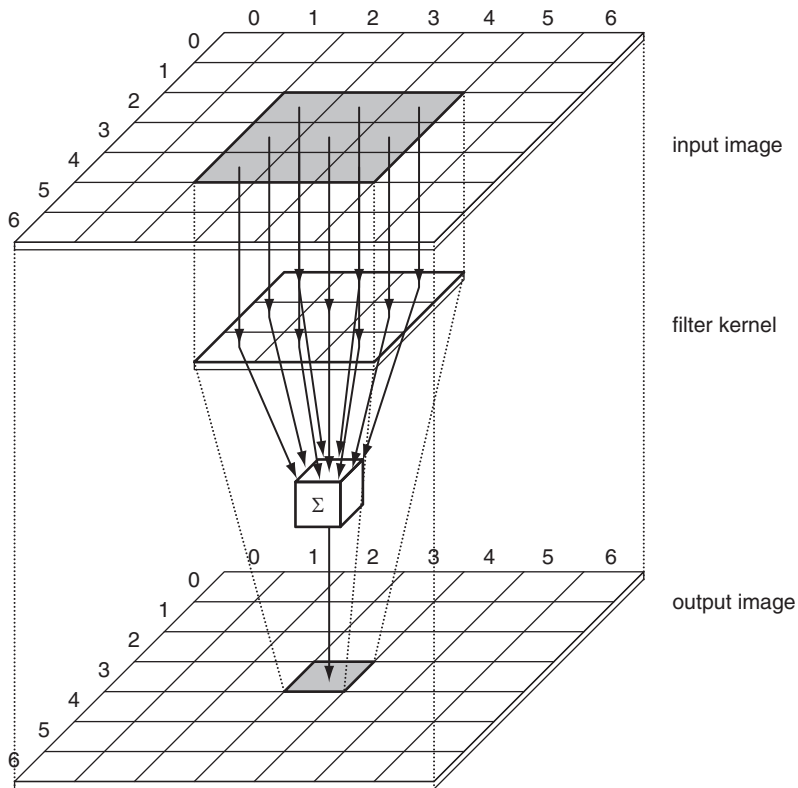


Fig. 2.8 Principle of a linear filter

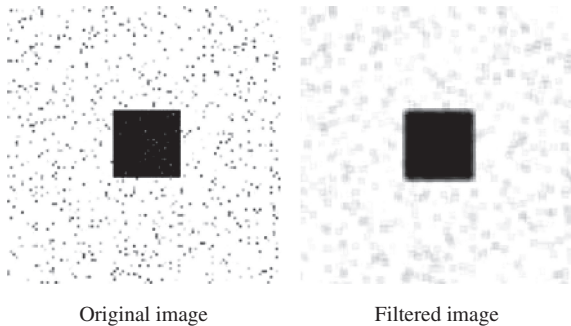
Since all filter weights are set to 1, Eq. (2.11) can be written for target pixel (3, 3):

$$\begin{aligned}
 g(3, 3) &= \frac{1}{9} \sum_{r=0}^2 \sum_{c=0}^2 g_i(3-1+c, 3-1+r) \\
 &= \frac{1}{9} (g_i(2, 2) + g_i(2, 3) + g_i(2, 4) + g_i(3, 2) + g_i(3, 3) + g_i(3, 4) \\
 &\quad + g_i(4, 2) + g_i(4, 3) + g_i(4, 4)).
 \end{aligned}$$

You can see that the result is simply the average of the gray values of all nine pixels within the filter kernel. All linear filters work in this way, but by using different filter weights, very different effects can be achieved.

Border effect: Image borders present a problem when using filters. When you calculate the output pixel (0; 0) using a square filter of size 3×3 , five of the required nine pixels are missing since they would have negative coordinates. The

Fig. 2.10 Effect of a 3×3 rectangular filter



2.3.3 Smoothing Filters

Low-pass filters used to suppress noise or tiny details are generally called smoothing filters. They are based on averaging. Since noise is nothing but randomly changing gray values of individual pixels, it can be suppressed by averaging brightness within small image areas. If there is, for example, a dark noise pixel in a light area, the brightness of this area will decrease slightly as a result of the averaging. The gray value of the dark noise pixel will be increased distinctly because of the surrounding light pixels.

Rectangular filter: The most simple smoothing filter is the rectangular filter. It is also often called mean value filter because it is equivalent to computing a statistical average: all gray values within the operator are added and divided by the number of operator pixels. The scaling factor f_{cr} from Eq. (2.11) is set to 1 and can therefore be left out.

We will use this simple example to illustrate the functioning of a linear filter once more in detail since you will encounter this principle with many other types of filters. For this we will use, as in Eq. (2.12) and Fig. 2.8, a rectangular filter with a size of 3×3 , i.e. the filter kernel is defined as follows:

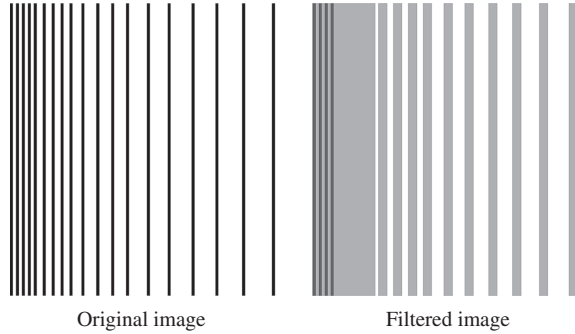
$$\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \quad (2.12)$$

The conversion for a pixel follows from Eq. (2.11) as:

$$g(x^*, y^*) = \frac{1}{9} \sum_{r=0}^2 \sum_{c=0}^2 f_{cr} g(x^* - 1 + c, y^* - 1 + r). \quad (2.13)$$

Figure 2.10 shows the effect this filter has on the image of a black square against a white background to which noise has been added. You can see that the noise is less distinct: individual noise pixels are spread out, but they stand out less from the white background than they do in the original image. You can also see

Fig. 2.11 Effect of a 3×3 rectangular filter on periodic structures



that the sharp edges of the square have spread out. They now consist of several lines decreasing in brightness from the outside to the inside.

Edge blurring can best be seen in an image with periodic structures as in Fig. 2.11. Processing this figure will demonstrate additional properties of this simple filter:

- The widely separated lines on the right side of the image remain lines, but they are spread out and become lighter. The position of the lines cannot be located as precisely after filtering as it could before.
- Lines whose distance from one another equals the width of the filter kernel merge completely. This corresponds to the boundary wavelength of the filter.
- Lines whose distance from one another is less than the width of the filter kernel are not completely suppressed. Their amplitude is only lowered by a factor of three, i.e. from 255 to 85. The following calculation, which can be extended to both sides according to image dimensions, shows this explicitly. Thus, you can see that the blocking effect of this low-pass filter is not particularly good. In Jähne (2005) these properties are shown using the calculation of frequency responses with the help of a Fourier transform.

$$\begin{array}{ccccccc}
 \vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots \\
 \dots & 255 & 0 & 255 & \dots & * & \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} & = & \dots & 170 & 85 & 170 & \dots \\
 \dots & 255 & 0 & 255 & \dots & & & & \dots & 170 & 85 & 170 & \dots \\
 \dots & 255 & 0 & 225 & \dots & & & & \dots & 170 & 85 & 170 & \dots \\
 \vdots & \vdots & \vdots & & & & & & \vdots & \vdots & \vdots & &
 \end{array}$$

Figure 2.12 illustrates the disadvantages of this very simple kind of filter with the help of a bar code. The edge destruction caused by noise makes reliable identification of the bar code impossible. After filtering, the noise will be subdued, but because of the blurred edges, the intervals are so drastically reduced that reliable identification becomes impossible. This effect increases the larger the filter kernel becomes.

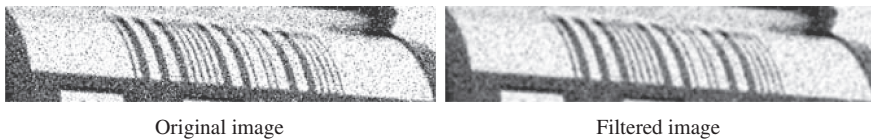
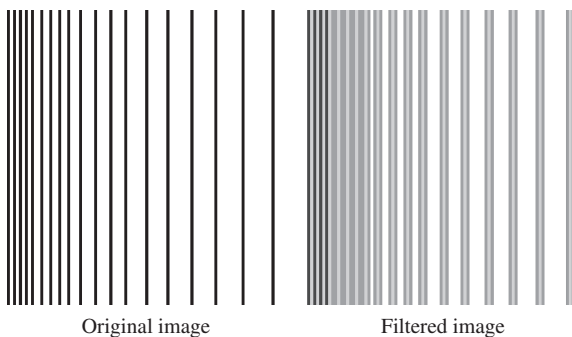


Fig. 2.12 Effect of a 3×3 rectangular filter on a noisy bar code

Fig. 2.13 Effect of a sparse 3×3 rectangular filter on periodic structures

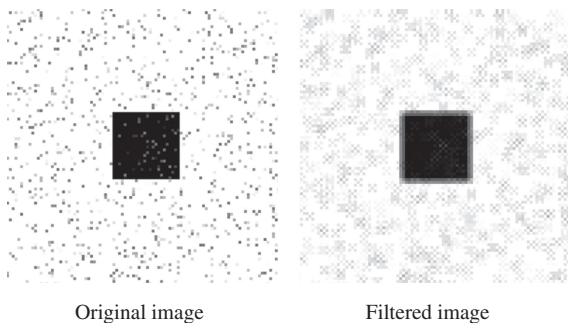


Sparse rectangular filter: Image filtering is computation-intensive and time consuming. However, roughly half the number of kernel points can be left out without significantly diminishing the filter effect. To keep the form of Eq. (2.11) the kernel points omitted are written as zeroes. During execution these kernel points are, of course, simply skipped. A sparse 3×3 filter kernel therefore looks as follows:

$$\frac{1}{5} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}. \quad (2.14)$$

The effect on linear, periodic structures changes considerably as you can see in Fig. 2.13. The filter is even less effective for lines that are close together than with

Fig. 2.14 Effect of a sparse 3×3 rectangular filter on a solid object



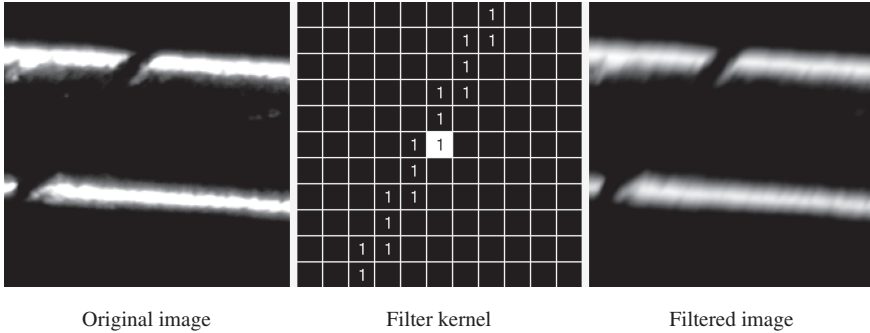


Fig. 2.15 Effect of a specially adapted mean value filter

the full 3×3 filter; the complete suppression of lines with a distance from one another of three pixels does not take place either. Because of the diagonal grid structure of the filter, lines that are further apart have a double-edged appearance.

Despite the sparse filter effect on periodic structures, sparse filters like this can be successfully used for “normal” image scenes because of the high autocorrelation of real-world scenes, i.e. neighboring pixels usually do not differ so much. The autocorrelation of line images, however, is very small, since moving the lines that are only one pixel apart for a distance of one pixel causes a major change in correlation. You achieve the same filter effect with half the computation time for the rather wide objects of real-world scenes except for the imprint of a slight texture caused by the diagonal grid structure of the filter. Figure 2.14 shows this with the help of the black square used above.

Mean value filters with customized operators: It is only a small step from the reduction of the filter kernel achieved by omitting every other operator point to filter kernels that are specially adapted to specific image structures and applications. Figure 2.15 shows the effect of a filter kernel that was designed to smooth edges running through an image at a certain angle. The object to be inspected is a guitar string whose metal coating is defective. To measure the distance between the windings, the average distance between the edges of the light objects has to be measured. This is much easier when these edges are as smooth as possible. Therefore, a filter kernel was designed that runs at the same angle the edges do. The filter kernel has an overall dimension of 6×11 pixels. The target pixel is indicated by a white frame.

Program hint: In **NeuroCheck** you can design adapted operators for every linear and many non-linear filters by selecting the “User-defined” filter in function *Filter Image*. You can select one of the standard filters as a template for the operator and modify it according to your needs.

Gaussian filter: Using the Fourier transform of a transfer function, one can show that an optimal smoothing filter has to have the shape of a Gaussian normal curve. In the spatially discrete and quantized space of a digital image, this function is approximated using integral filter weights. Equation (2.15) shows Gaussian filter

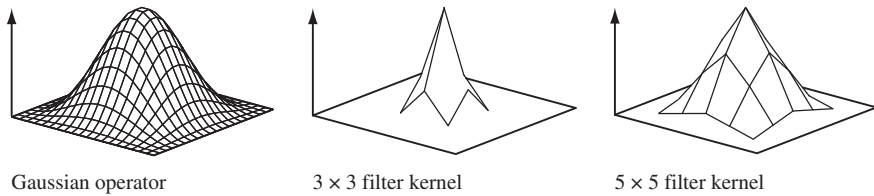
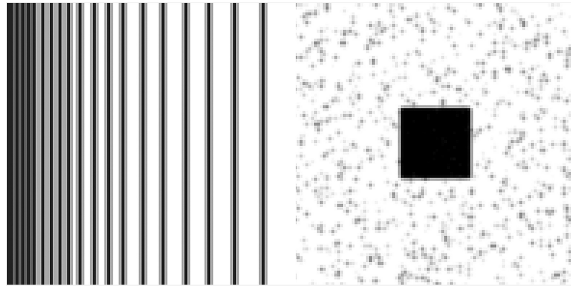


Fig. 2.16 Gaussian filter operator and discrete approximations

Fig. 2.17 Effect of a 3×3 rectangular filter on periodic structures and noisy objects



kernels with the dimensions 3×3 and 5×5 . Larger kernels can be found in Russ (2007), but remember that even with these optimized filters, a larger operator causes increased blurring and computation time increases with the area of the operator.

$$\begin{aligned}
 G_3 &= \frac{1}{28} \begin{pmatrix} 1 & 4 & 1 \\ 4 & 12 & 4 \\ 1 & 4 & 1 \end{pmatrix}, \\
 G_5 &= \frac{1}{121} \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix}.
 \end{aligned} \tag{2.15}$$

Figure 2.16 shows a theoretical Gaussian operator and the approximation using two filter kernels from Eq. (2.15). The smaller the filter kernel, the bigger the error due to discretization.

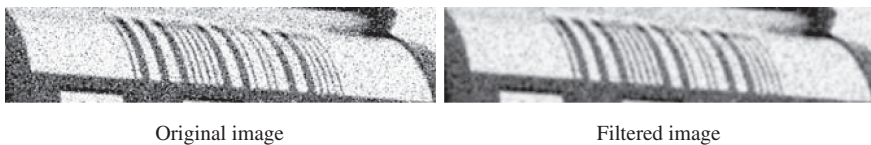


Fig. 2.18 Filtering of a noisy bar code with a 5×5 Gaussian filter

Figure 2.17 shows the effect of a Gaussian filter (3×3) on the examples from Figs. 2.10 and 2.11. Of course, edges are blurred, but they do not merge completely, and, using adequate interpolation models, the edge positions can still be located.

Figure 2.18 shows the filtering of the noisy bar code from Fig. 2.12 with a 5×5 Gaussian filter. Edges remain sufficiently sharp to identify the bar code.

Filter methods of this kind are extremely time consuming. Jähne (2005) gives a number of possibilities to use filter properties to speed up computation. These methods usually cannot be employed for adjusted filters since properties such as symmetry and separability into horizontal and vertical operations can be lost when the operator shape is altered.

Program hint: In **NeuroCheck**, the computation of the filtered image in the check function *Filter Image* is automatically distributed over all available processor cores for each linear and non-linear type of filter.

2.3.4 Edge Filters

Low-pass filters smooth images by suppressing variations. Edges are nothing but variations, i.e. fast changes of the gray value in a confined space. This suggests the use of high-pass filters to set off edges. This is indeed possible using difference filters. Taking the most simple case, the variation of a function is determined numerically by calculating the difference of neighboring function values. Using this principle, edge filters are designed that have to contain negative filter weights to compute differences. This presents us with the same problem we had with image subtraction in Sect. 2.2.2: the resulting values cannot only be bigger than the highest gray value—as is the case with smoothing filters—they can also be negative. It is therefore not necessarily sufficient to use the simple scaling method of division by the sum of filter weights to map the results into the normal gray value range.

Principle of an edge filter: We will show the principle using a simple example with the following filter kernel:

$$\frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}. \quad (2.16)$$

The effect of this filter is a numerical approximation of the gray value gradient in X-direction. We will deal with the problem of possible negative values later. For the moment we will use the absolute values of the results. Scaling to the gray value range of $[0; 255]$ is achieved by using a standardization factor of $1/4$ in Eq. (2.16).³

³ We differ here from other literature that uses a scaling factor of $1/8$. We will demonstrate that $1/4$ is sufficient to assure a limitation to the gray value range.

Fig. 2.19 Effect of a simple edge filter

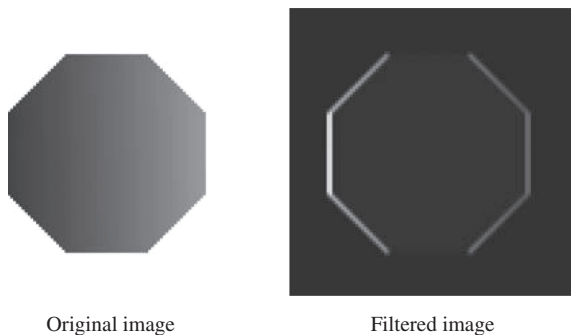


Figure 2.19 shows the result of filtering an image of a simple object using this operator.

The following properties can be recognized:

- Negative and positive components cancel each other in a completely uniform background, so the result is 0. The background turns uniformly black. The following equation shows the result of the computation of an individual background pixel.⁴

$$\begin{vmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{vmatrix} \times \frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} = 0.$$

- With perfectly horizontal edges, there is no change in the direction of differentiation of the filter. Therefore, pixel-wise multiplications cancel each other and horizontal edges disappear, as the following equation shows:

$$\begin{vmatrix} 255 & 255 & 255 \\ 128 & 128 & 128 \\ 0 & 0 & 0 \end{vmatrix} \times \frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} = 511 - 511 = 0.$$

- With perfectly vertical edges, the filter effect is maximal since every row of the filter operator contributes to the final result. The following equation shows this using an edge that ascends from the gray value 0 (black) via an intermediate step to the gray value 255 (white).

$$\begin{vmatrix} 0 & 128 & 255 \\ 0 & 128 & 255 \\ 0 & 128 & 255 \end{vmatrix} \times \frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} = \frac{1}{4} |-1020| = 255.$$

- This example shows why scaling with 1/4 is sufficient. It is impossible to get a higher absolute value than 1,020 in a gray level image using this filter kernel.

⁴ Please note that this is not a matrix multiplication where the scalar products of rows and columns are computed. Multiplication is done pixel by pixel.

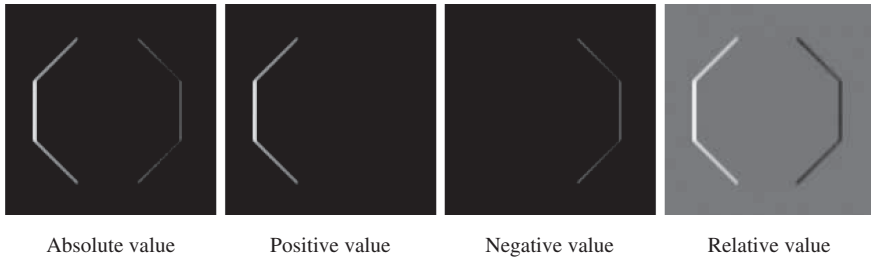


Fig. 2.20 Effect of different scaling modes for edge filters

Whenever a gray value *in front of* the edge increases, the absolute value decreases, since additional positive products are added; each decrease *behind* the edge decreases the absolute value too, since the negative products become smaller.

- With diagonal edges, the contribution of the first and last rows of the filter operator oppose each other and thereby weaken the effect, as the following equation shows:

$$\begin{vmatrix} 0 & 0 & 128 \\ 0 & 128 & 255 \\ 128 & 255 & 255 \end{vmatrix} \times \frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} = \frac{1}{4} |-765| = 191.25.$$

- The object is darker on the left than on the right, i.e. the edge is steeper on the left. In the output image created by the filter, the edges in the front are lighter than the ones in the back of the filter direction.

Scaling methods: Now we come to the question of how to deal with negative output values of such difference filters. Similar to image subtraction, one can distinguish between several scaling modes whose effects are shown in Fig. 2.20 using the example image from Fig. 2.19.

The different scaling modes have the following effect:

Absolute value: The final gray value is calculated by scaling the absolute value of the original result linearly to the gray value range. Information about edge direction is lost in the process since it is impossible to discern what sign the original result had.

Positive values: Only positive results are used and scaled linearly to the gray value range. This method effectively chooses a specific edge direction. Which edge is maintained depends on the filter kernel; in the example, the edges changing from light to dark were chosen.

Negative values: Only negative results are used, all positive results are set to 0. From the remaining negative values, the absolute values are used and scaled to the gray value range. Edges are maintained with a direction opposite to that for positive values.

Relative values: Here, all results are first scaled to a range symmetrical to zero, e. g. $[-128; 127]$, then moved to the range $[0; 255]$ by addition. All areas without image variations are thereby mapped to a medium gray value and edges of different directions can be distinguished depending on whether they appear light or dark in the output image.

Depending on the scaling method, Eq. (2.11) has to be changed accordingly to obtain the final gray value in the resulting image. However, this does not change the principle.

In the literature you will find an immeasurable variety of edge filters, only some of which can be mentioned here.

Laplacian operator: Unlike the simple operator from Eq. (2.16), Laplacian operators capture all edge directions simultaneously. Equation (2.17) shows two Laplacian operators of different size. You will also find these operators with opposite signs, which does not make any difference since they do not depend on direction.

$$L_3 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix},$$

$$L_5 = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}. \quad (2.17)$$

Figure 2.21 shows the effect of filtering by Laplacian operators and scaling by absolute value. Since these filters do not depend on direction, other scaling methods do not make much sense in this case.

Sobel operator: The Sobel method for edge extraction uses a series of coordinated filter operations that extract edges in different directions. A typical combination can be found in Sonka et al. (2008) (as in Eq. (2.16), we scale by $1/4$ and not $1/8$):

Fig. 2.21 Edge extraction using Laplacian filters (contrast enhanced to $[0; 255]$)



3 x 3-Filter

5 x 5-Filter



Fig. 2.22 Edge extraction using different Sobel operators and combination by a maximum operator

$$\begin{aligned}
 S_1 &= \frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, & S_3 &= \frac{1}{4} \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}, \\
 S_2 &= \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, & S_4 &= \frac{1}{4} \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}.
 \end{aligned} \tag{2.18}$$

Figure 2.22 shows the result of filter operations using these operators and scaling by absolute values. If you want to determine all edge directions, you can later combine the individual filtered images using a maximum operator according to Sect. 2.2.3. Since all edges are set off brightly, they are all contained in the maximum image.

Sobel operators of greater dimensions than 3×3 are of course also possible.

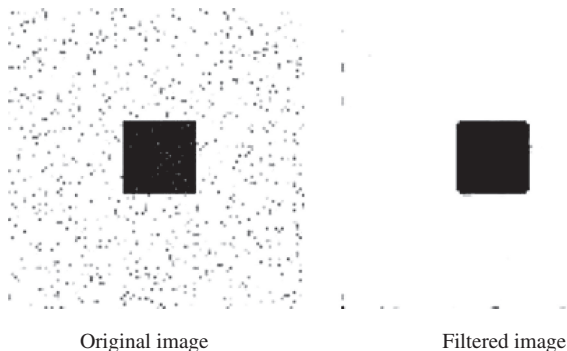
Sobel magnitude filter: We mention this filter here because of its close relationship to the Sobel operators, though it is not a linear filter but a non-linear combination of two Sobel filters. By computing the absolute value of the gradients in two directions orthogonal to each other, it achieves about the same result as a combination of four Sobel filter operations with the maximum operator. The image is filtered using two Sobel operators orthogonal to each other, such as S_1 and S_2 from Eq. (2.18). The absolute value of the derivation in both directions is computed. Filtering with S_1 determines an approximation for the gray value gradient in x-direction, i.e. $(\partial g / \partial x)$; filtering using S_2 gives the respective result in y-direction. The result of a Sobel magnitude filter is then equivalent to

$$\text{Magnitude} = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2}. \tag{2.19}$$

2.4 Median Filter

We saw that taking the average value, as in the simple mean value or rectangular filter, results in severe edge blurring. At the edge, intermediate values of the actual gray values in the image are generated. Furthermore, noise pixels, as we have seen

Fig. 2.23 Effect of a 3×3 median filter on a solid object



in Fig. 2.10, can cause significant changes in the average gray value of their neighborhood. The median is used in statistics to reduce sensitivity to outliers. The median is equivalent to the central value of a set of ordered values, in this case the set of gray values given by the filter operator. If the set of gray levels represented by the filter kernel is ordered by value,

$$g_1 \leq g_2 \leq \dots \leq g_n,$$

the median is, according to Bronstein et al. (2005), defined as

$$\bar{g} = \begin{cases} g_{\frac{n+1}{2}} & , \text{ for odd } n \\ \frac{1}{2}(g_{\frac{n}{2}} + g_{\frac{n}{2}+1}) & , \text{ for even } n. \end{cases} \quad (2.20)$$

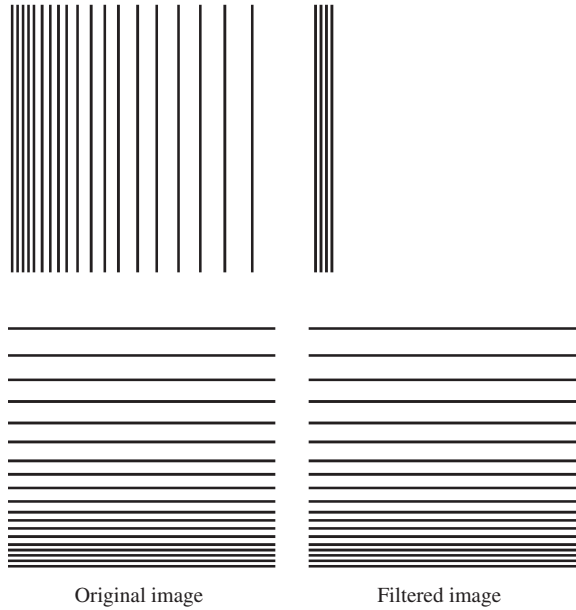
The median filter is a frequently used special case of rank filtering. Rank filters are based on what is called quantiles in statistics. A quantile x_q is defined such that a fraction q of the values of a discrete series are less than or equal to x_q . According to this definition, the median is an $x_{1/2}$ quantile. Generally you can use any quantile to achieve various effects; however, the median filter is the most frequently used.

The median filter is ideally suited to eliminate pixel noise while maintaining edge sharpness. Due to the necessary sort operation, the median filter is much more time-consuming than a mean value filter of the same size.

Figure 2.23 shows the effect of a 3×3 median filter on a noisy image. You can see that almost all noise spikes were eliminated, whereas edge sharpness was maintained—except for some minor irregularities.

A dense 3×3 median filter orders nine values from a square neighborhood of each pixel into a series and uses the fifth value as the gray value of the pixel in the output image. Just like a mean value filter, a median filter can take any shape. Using a horizontal 3×1 median filter, vertical lines with a width of one pixel can be completely eliminated if they are sufficiently far apart from each other, whereas horizontal lines are not affected as can be seen in Fig. 2.24.

Fig. 2.24 Effect of a 3×1 median filter on one-dimensional structures



2.5 Morphological Filters

The preprocessing methods discussed so far primarily used methods from matrix algebra and calculus. Another class of non-linear filters, however, is based on mathematical morphology. Morphology means literally “the science of shapes”. Mathematical morphology is based on set-theoretical concepts as shown in Gonzalez and Woods (2008) and Sonka et al. (2008).

For practical application in image preprocessing, morphological filters are usually implemented in the form of minimum and maximum operations. According to Nischwitz et al. (2007), basic morphological operations such as *erosion* and *dilation* assign the minimal or maximal gray value respectively from within the filter kernel to the output pixel.

Erosion: If we denote the neighborhood given by the filter kernel as N^* , as for the local operation in Eq. (2.10), the resulting gray value g_e for the erosion of the gray values g_i of the source image is obtained by:

$$g_e(x^*, y^*) = \min\{g_i(x, y) | (x, y) \in N^*\}. \quad (2.21)$$

The term erosion stems from the effect that this operation thins out light structures. As soon as there is a single dark pixel within the filter kernel, the central point becomes dark as well. A solid light object is consequently eroded all around. On the other hand, you could also say that an erosion thickens dark structures. Figure 2.25 shows the effect of a 3×3 erosion filter. Every noisy pixel is increased in size to match the size of the kernel. The central square is enlarged too,

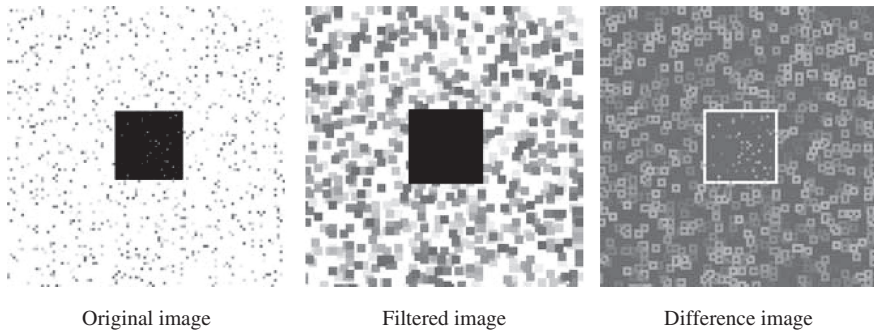


Fig. 2.25 Effect of a 3×3 erosion filter

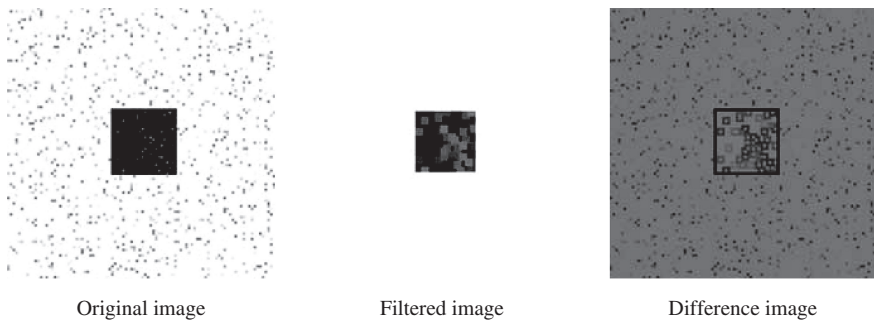


Fig. 2.26 Effect of a 3×3 dilation filter

which can be seen from the fact that a white frame remains when subtracting the filtered image from the source image (subtraction was performed using relative scaling, i.e. all pixels that are identical in both images are set to a medium gray value; cf. Sect. 2.2.2).

Dilation: Dilation is the complementary operation to erosion, i.e. the resulting gray value g_d of the dilation is obtained by:

$$g_d(x^*, y^*) = \max\{g_i(x, y) | (x, y) \in N^*\}. \quad (2.22)$$

Please note that although dilation is by definition the complementary operation of erosion, the operations cannot be reversed. Performing a dilation after an erosion does not restore the original image. The same is true for performing an erosion after a dilation. Both combinations are distinct operations which we will discuss in detail a little later. The effect of a dilation using a 3×3 kernel can be seen in Fig. 2.26. The noise pixels have completely disappeared since there always is a light background pixel in their neighborhood. The size of the black rectangle has been reduced accordingly, which is shown by the dark frame in the difference image.

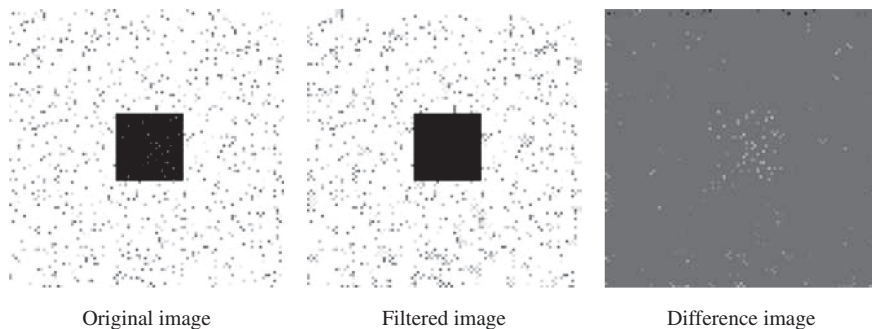


Fig. 2.27 Effect of a 3×3 opening filter

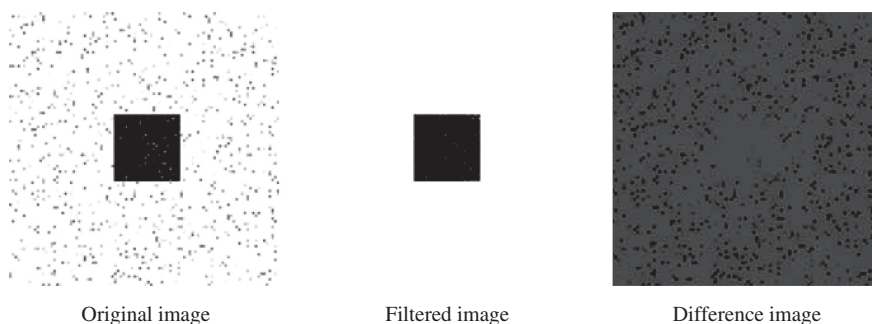


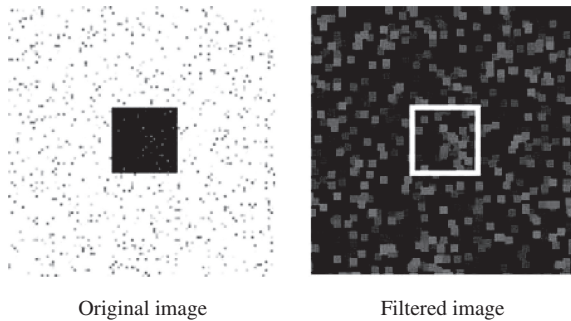
Fig. 2.28 Effect of a 3×3 closing filter

Obviously, dilation causes an increase in width of light structures and a decrease in width of dark structures. This operation is therefore ideally suited to prevent breaking up of light characters as we will see in Sect. 5.3.

Opening: *Opening* denotes the combination of an erosion followed by a dilation. Erosion closes gaps in dark structures whereas dilation reverses the thickening of dark objects. The enlargement of dark areas because of erosion is thus canceled by dilation. If the gaps in the dark structures were so small compared to the filter kernel that they were completely closed and no light pixel was left, they cannot be reopened by dilation. The term opening stems from the fact that thin light structures are broken up using this operation while object features such as area and basic dimensions are retained. Figure 2.27 shows the effect of an opening operation. The light noise pixels within the square were removed but the dimensions of the square did not change. Therefore, there is no square frame in the difference image.

Closing: *Closing* is the complementary operation to opening: first, a dilation is performed followed by erosion. The effect is also reversed: gaps in light objects are closed without change in the object area (hence the name). Figure 2.28 shows this effect using a dark square on a bright background, both covered with noise.

Fig. 2.29 Effect of a 3×3 contrast filter on a solid object



Dilation overwrites the dark noise pixels within the background with white pixels, at the same time shrinking the dark square. A subsequent erosion reverses the change in the dimensions of the square so that there is no frame in the difference image.

2.6 Other Non-linear Filters

There are a number of other techniques used to improve image quality. The most effective of these use a priori knowledge about the kind of image defect, for example Wiener filters as described in Rosenfeld and Kak (1982), Sonka et al. (2008), that assume that the statistical parameters of the noise are known. Under these conditions, one can perform remarkable feats of image restoration; however, this information is not usually known. Other methods rely on the fact that neighboring lines or gaps usually have a high correlation. That way, image rows lost because of transfer difficulties, e.g. data sent by deep space probes, can be restored. However, one has to keep in mind that this is only a visual improvement. The lost information cannot actually be restored. Furthermore, these methods are usually too computation-intensive for industrial applications.

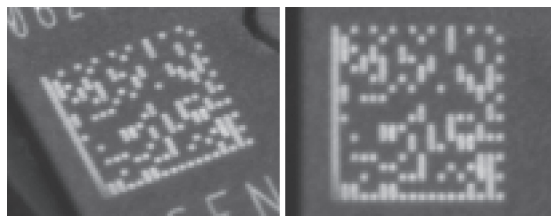
Finally, we want to mention the min-max or contrast filter. It assigns to each pixel in the filtered image the difference of the smallest and largest gray value of the original image within the filter kernel. The effect of this filter is to enhance variations within the image and further homogenize relatively uniform areas. Figure 2.29 shows this effect using our familiar example square. You can see that the noise is enhanced and thickened, but not as much as the edges of the square. At the same time, one notices an “outlining effect”. Since the inside of the square is uniform—except for the noise—an empty square remains.

The advantage of this filter for certain applications is that the direction of the variation is not important, i.e. a light spot against a dark background and a dark spot against a light background look identical after using a contrast filter. We will use this effect in Sect. 5.3 to equalize the different reflection properties of stamped digits whose individual points are not of the same depth. Figure 2.30 gives a



Original image

Filtered image

Fig. 2.30 Stamped digits before and after contrast filtering**Fig. 2.31** Coordinate transformation on a DataMatrix code

Original image

Rectified image

preview of this application. In one of the original images, the digits appear light, while appearing dark in the other. After applying a difference filter, contrast properties are almost identical.

2.7 Coordinate Transformations

With the preprocessing methods presented so far, the gray values of the pixels were changed from the input to the output image while the arrangement of pixels with regard to each other remained unchanged. The coordinate transformations we are going to present in this section work the other way around: while the gray values of the pixels remain unchanged, the coordinates are changed. Hence, the pixels in the output image are in a different position from the input image.

Coordinate transformations are used when a test piece is captured from an unfavorable angle and when the image has to be rectified, i.e. perspective distortions need to be removed. Figure 2.31 shows a DataMatrix code captured from an unfavorable angle before and after rectification. Even with this kind of preprocessing, one should stick to the rule to optimize the image capturing situation before trying to rectify errors with preprocessing algorithms.

A coordinate transformation can be described by

$$\begin{aligned}x_o &= f_x(x_i, y_i), \\y_o &= f_y(x_i, y_i).\end{aligned}\tag{2.23}$$

With (x_i, y_i) as the coordinates of the pixels in the input image and (x_o, y_o) as the coordinates in the output image. The following relation holds for the gray values in the input image g_i and the output image g_o :

$$g_o(x_o, y_o) = g_i(x_i, y_i).\tag{2.24}$$

To compute the gray value of an output pixel, the coordinates of the corresponding input pixel have to be calculated and the gray values adopted. In most cases, the input pixel corresponding to a specific output pixel is not located exactly on one pixel but somewhere among four pixels. Therefore, the coordinates of the relevant pixel usually do not have integer values. To compute the gray values for these in-between pixels, there are a number of common interpolation methods such as pixel-repetition and bilinear interpolation.

Pixel-repetition: For pixel-repetition the gray value of the pixel closest to the computed coordinate is used. It can happen that an input value is adopted several times in the output image, hence the name of the method. Pixel-repetition is the simplest and thus quickest method, however, it results in poor output image quality. Especially when there are sloped edges in the images, characteristic steps appear.

Bilinear interpolation: For bilinear interpolation the output value from the gray values of the four input pixels situated around the theoretical input pixel is interpolated. Interpolation takes place both in x-direction and y-direction, hence the name bilinear. If the computed point (x, y) is located between the four pixels (x_1, y_1) , (x_2, y_2) , (x_3, y_3) and (x_4, y_4) with the gray values g_1, g_2, g_3 and g_4 , the sought gray value is the weighted mean value of the four neighboring pixels according to the following equation:

$$g = \sum_{i=1}^4 (1 - |x_i - x|) (1 - |y_i - y|) g_i.\tag{2.25}$$

Since the distance of the pixel grid is 1 and the computed pixel is between the pixels on the grid, the values are always between 0 and 1. The weight that is attributed to the gray value of a pixel in the sum is greater the closer the computed pixel is to the output pixel.

To be able to do the coordinate transformation, f_x and f_y from Eq. (2.23) are still missing. According to Jähne (2005), perspective distortion can be described using

$$\begin{aligned} f_x(x, y) &= \frac{ax + by + c}{gx + hy + 1}, \\ f_y(x, y) &= \frac{dx + ey + f}{gx + hy + 1}. \end{aligned} \quad (2.26)$$

To rectify the image, the coefficients from Eq. (2.26) have to be determined and the inverse function computed. The coefficients are computed by determining the coordinates of at least four pixels in the input image and establishing the corresponding output image coordinates. From these pairs of coordinates the coefficients are computed either for four exact pixels, or, if there are several pairs of coordinates, by fitting. For a detailed description of coefficient computation please refer to Jähne (2005).

There are a number of extensions and alternatives to the methods described. An often used extension of rectifying perspective distortions is the correction of lens distortions. In other literature these methods are often described together with transforming images into world coordinates as camera calibration, cf. (Steger et al. 2008). Besides rectification with fixed functions as described above, you can also use two-dimensional spline functions that need more control points, i.e. pairs of coordinates in input and output image, but can also correct more diverse distortions.

2.8 Integral Transformations

At the end of this chapter about image preprocessing, we would like to mention that there are a number of global image transformations based on integral transformations we will not go into here for various reasons. Among these are the Fourier transform and the family of wavelet transforms. Both methods can be used to describe textures, i.e. surface structures, and to describe objects with invariance to translation and rotation.

We do not discuss these methods for the following reasons:

- They are quite computation-intensive, since a multitude of operations has to be performed on the whole image.
- The mathematics needed for understanding the Fourier transform and the interpretation of its results go far beyond the scope of this introductory book.
- Texture analyses of this kind are not typical tasks in industrial vision.
- Object features based on Fourier and wavelet transforms are mostly used for handwritten character recognition. For industrial purposes, translation and rotation can usually be compensated for using less complex methods.

If you are interested in these topics in more detail, please refer to Jähne (2005), Tönnies (2005), Nischwitz et al. (2007), Gonzalez and Woods (2008), Russ (2007), and others.

2.9 Summary

Image preprocessing: In image preprocessing a new image is generated from a source image; the new image is similar to the source image, but differs from it with regard to certain properties, e.g. having better contrast or less noise.

Pixel operations: This type of preprocessing operations uses only the pixel from the corresponding spot of the input image(s). For inhomogeneous pixel operations, the transformation function depends on the location of the pixel in the image, for homogeneous operations it does not. The most important pixel operations are gray scale transformations and image arithmetic.

Gray scale transformation: This is an homogeneous pixel operation which replaces the gray level i_{ij} of every pixel in the input image by a new gray level o_{ij} using a mapping function $f(i_{ij})$. The mapping function does not depend on the position of the pixel in the image.

Look-up tables: A look-up table contains the result of an arbitrary mapping for every possible input gray level so that the result of the transformation can be looked up directly without actually performing the mapping for each pixel. This can help to speed up many transformations such as logarithmic functions. Also, analytically impossible transformations can be described like this.

Linear gray level scaling: For this transformation, the gray level is transformed according to equation $o_{ij} = c_2 \cdot (c_1 + i_{ij})$.

Contrast normalization: By an appropriate choice of the coefficients of a linear gray level scaling an image can be transformed such that it uses the full gray level range of [0; 255]. If the image is subdivided into smaller regions prior to the transformation, a *local contrast normalization* can be achieved. Strictly speaking, this algorithm is not a homogeneous pixel operation, but it can be implemented in a similar way. Local normalization can be used to compensate for illumination gradients.

Histogram equalization: This operation transforms the gray scale such that all brightness intervals are equally occupied. It is used primarily to improve recognition of slight gray level contrasts by humans.

Image arithmetic: Denotes operations using corresponding pixels from two or more source images. The most important operations are:

- Adding and averaging, e.g. for noise suppression in dark image scenes
- Subtracting, e.g. for print quality inspection and shading correction
- Minimum and maximum, e.g. to combine the resulting images from edge filtering operations
- Dividing, e.g. for shading correction.

Most arithmetic operations with images can create results outside the gray level interval $[0; 255]$. Various methods (clipping, linear scaling, scaling of positive, negative or absolute values) can be used to map these results to the standard gray level interval for further processing or display.

Local operations: This type of preprocessing operation derives the gray level of a pixel in the new image from a neighborhood N^* of the corresponding pixel in the source image. This type of operation is also called image filtering with the neighborhood N^* as the *filter kernel* or *filter mask*.

Linear filter: A type of local operation which computes the new gray level as the weighted sum of the gray levels in the neighborhood N^* of the corresponding point in the source image. Examples are smoothing filters and edge filters. With increasing kernel size, the filter effect generally becomes stronger, while the computation effort also increases linearly with the number of points in the neighborhood.

Smoothing filters: These filters are used mainly for noise suppression. The most basic type is the *rectangular filter*, often also called *mean value filter*. The *Gaussian filter* achieves better noise suppression while simultaneously causing less blurring of edges.

Edge filters: These filters basically perform a numerical differentiation to emphasize variations in the image. They are usually directionally specific, e.g. operating on horizontal or vertical edges only. Common edge filters are the *Sobel operators* and the *Laplacian operators*, but there are many others.

Median filter: This is a non-linear filter, because it uses the more robust median of the gray levels in the neighborhood N^* instead of the mean value employed by the basic smoothing filter. It causes less edge blurring than mean value filters and is best suited for the suppression of pixel noise.

Morphological filters: This type of filter affects the shape of image structures, typically through minimum and maximum operations. They are therefore non-linear filters. The most common morphological filters are *erosion*, *dilation*, *opening* and *closing*.

Contrast filter: This non-linear filter replaces the gray level of every pixel by the maximum contrast inside the neighborhood N^* . The effect is that light and dark variations are brightly marked in the resulting image, whereas more uniform areas will appear dark.

Coordinate transformations: Coordinate transformations change the arrangement of pixels in an image. This can be used, e.g. to correct images distorted because of unfavorable camera perspectives.

Global operations: Global preprocessing algorithms use the entire source image to compute the new image. Common examples are integral transforms like the Fourier transform or the wavelet transform. They facilitate very powerful and complex image manipulations while being computationally rather expensive.

References

- Bronstein IN, Semendjajew KA, Musiol G, Mühlig H (2005) Taschenbuch der Mathematik. 6th edn. Harri Deutsch, Frankfurt a. M
- Gonzalez RC, Woods RE (2008) Digital image processing, 3rd edn. Pearson Education, Upper Saddle River
- Jähne B (2005) Digitale Bildverarbeitung, 6th edn. Springer, Berlin
- Nischwitz A, Fischer M, Haberäcker P (2007) Computergrafik und Bildverarbeitung, 2nd edn. Vieweg, Wiesbaden
- Rosenfeld A, Kak AC (1982) Digital picture processing, 2nd edn. Academic Press, New York
- Russ JC (2007) The image processing handbook, 5th edn. CRC Press, Boca Raton
- Sonka M, Hlavac V, Roger B (2008) Image processing, analysis, and machine vision, 3rd edn. Cengage Learning, Stamford
- Steger C, Ulrich M, Wiedemann C (2008) Machine vision algorithms and applications. Wiley-VCH-Verlag, Weinheim
- Tönnies KD (2005) Grundlagen der Bildverarbeitung. Pearson Studium, München

Chapter 3

Positioning

It might appear unusual to start a detailed discussion of the various application areas of industrial image processing with positioning instead of object recognition since an object must first be found before its position can be determined. However, object recognition is a rather broad term and frequently necessitates a multitude of functions to be able to assign objects to a category. Positioning, however, is structurally speaking—not necessarily algorithmically—a rather simple affair as soon as the object in question has been found. The only necessary prerequisite is the segmentation of a reference object. The main reason to begin with this topic is that it represents an absolutely essential “auxiliary science”.

The example in Sect. 1.6 was so easy to solve only because the exact location of the object to be tested was already known. Especially gauging—and even more so the described method for presence verification—rely on finding the item to be checked in the same position with very small variations, whereas type numbers might be found anywhere in the picture employing a time-intensive search. Although it can usually be presumed in industrial manufacturing that parts to be checked are positioned within certain limits, these limits are usually not narrow enough for the requirements of image processing. In most cases it is therefore unavoidable to determine the position of reference objects first, which are easy to find—and which are certain to be within the camera’s field of view—and then start the inspection itself.

3.1 Position of an Individual Object

Let us return to the example from Sect. 1.6. Even if the spark plug is attached to a work-piece carrier to be presented to the camera, one has to expect small variations in position in actual applications. These are especially inconvenient when there are only extremely small areas allowed in which to search for the object, as is the case in these inspection subtasks, i.e. presence verification and gauging.

Figure 3.1 shows the effect of such a position deviation on the search lines for gauging the thread depth from Sect. 1.6, shown here under front lighting.

Fig. 3.1 Search lines for thread contour on horizontally displaced spark plug



Neither of the search lines intersects the thread in the middle; correspondingly, the boundaries of the surrounding areas are wrong, so that the contour found has nothing to do with the thread contour to be measured.

Program hint: You can easily simulate such a displacement in one of the example programs from the download by moving the defined image section in function *Transfer Image*.

It is obvious how to solve the problem: at first, it has to be determined where the thread is positioned within the image and then move the search lines accordingly. Various strategies are conceivable and will be discussed in the following sections.

3.1.1 Positioning Using the Entire Object

The simplest way is to segment the whole spark plug as an object and to determine the position of these segments. During the first execution of this process, the determined position is stored as a reference and the search lines are defined according to this reference position. In all following executions, the difference between the present position and the stored reference is determined and the search lines moved accordingly. In this specific case it is of course sufficient to determine the deviation in the horizontal direction.

The solution of the inspection task itself is not at all different from the approach in Sect. 1.6, only the second step changes: the setting of regions of interest, since their positions can only be determined relative to the position of the whole spark plug. The whole process will be explained in detail in the following.

1. *Image capture*
2. *Set ROI to search for reference position:* The size and position of the region of interest to be searched for the reference position is determined by the type of reference object and the positioning uncertainty to be expected. In this example, the horizontal position varies only minimally, it is therefore sufficient to search a certain area on the left side of the image to find the transition between background and object as can be seen in Fig. 3.2.
3. *Create reference object:* After the transition between background and reference object has been found, the entire spark plug can be created as an object using the method of contour tracing described in Chap. 4, as can be seen in Fig. 3.3.

Fig. 3.2 Area to be searched to create the reference object

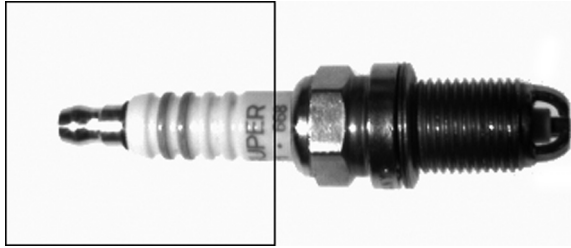


Fig. 3.3 Entire spark plug as reference object



4. *Compute object properties:* The only essential characteristic of this reference object is its position in horizontal direction. In this step therefore, the coordinates of the object's center of gravity are computed.
5. *Set ROI for inspection:* After the reference position has been determined, all necessary information is available to determine the final search areas, in this case 173 pixels to the right of the object's center of gravity. Figure 3.4 shows the search lines positioned relative to the object position from Fig. 3.1.

The rest then proceeds just like in the example in Sect. 1.6 (which is why only the sequence of the steps is listed).

6. *Create objects:* The edges of the thread are searched for on the search lines, which cut through the thread in the middle once more, and the contour lines are created within the edges.
7. *Compute object properties:* The minimum and maximum distance of the contour lines in vertical direction is determined.
8. *Decision:* If both distances are within the prescribed limits, the thread is declared to be faultless.

This procedure is still in keeping with the solution approach from Sect. 1.5, but there is a complete second process embedded in the setting of the ROI for the inspection, which determines the reference position.

Fig. 3.4 Search lines correctly positioned in relation to spark plug



Program hint: When configuring a check routine for use in manufacturing—or, more generally speaking, whenever an inspection has to be built based on real camera images—one should make a habit of storing (at least) one camera image as reference. This image should be used to configure a position correction so that the position reference is always available at a later date. For this, **NeuroCheck** allows you to provide the reference image with markings and compare it with the current camera image to detect positioning errors easily.

3.1.2 Positioning Using an Edge

The above mentioned method to use the center of gravity of the entire object as the reference position is fascinating because of its simplicity and is also preferable because of its robustness. Since an averaging over all pixels of the object is performed when computing the center of gravity, the position is relatively insensitive to small changes in illumination or focus. A possible disadvantage, however, is that the results can be misleading if the object's shape changes because of image capturing conditions.

Change of shape because of segmentation problems: Character recognition necessitates front lighting so that the characters to be read are visible. This presents a certain degree of risk that the segmentation will fail in part because of the light ceramic body and the shiny thread. As a consequence, the shape of the reference object changes, and this in turn causes a change in the position of the center of gravity. Figure 3.5 shows an extreme case in which the entire rear section of the object area is no longer segmented, which moves the center of gravity from its correct position, marked in the figure with “0”, to position “1”.

Such blatant errors as in Fig. 3.5 are of course rare and easy to discern, but more subtle deviations can even occur with back lighting. Too strong a reflection from the ceramic body can easily lead to a partial merging with the background, as can be seen in Fig. 3.6, which also causes errors in sensitive inspections.

Change of shape because of excessive movement: Resulting from a change in object shape, the movement that actually occurred is wrongly computed, as can be seen in Fig. 3.7. There, the left part of the object has moved out of the image and is no longer segmented.

In a case such as this it is advisable not to use the center of gravity for positioning but the position of an object edge. What is needed then is a vertical edge as x-reference and a horizontal edge as y-reference. Such an edge has to be wide enough so that it can be found despite position uncertainties in the other direction. This is not trivial in the case of the spark plug, especially with regard to the x-reference since

Fig. 3.5 Pseudo-shift of the center of gravity because of segmentation problems



Fig. 3.6 Partial background merging because of overexposure at the contour



Fig. 3.7 Pseudo-shift of the center of gravity because of an object movement over the image edge



there are hardly any vertical edges. Therefore, one would first search for a horizontal edge as a y-reference such as the upper edge in the area of the hexagon. If the chosen detected edge is wide enough, it is assured that the upper limit of the spark plug is always contained, so that a precise y-reference is obtained. Figure 3.8 shows the corresponding search area and the detected contour section.

Using this vertical position, new search lines can be positioned that can be used to find the short vertical edges of the hexagon as x-reference. Such a hierarchical procedure is frequently employed for position-critical inspections. The two position values found in the direction of x and y give faultless results in the two cases mentioned above, unlike the method that employs the center of gravity—moreover, it uses up less computation time.

Error sources using edge positioning: However, using an edge for positioning is no panacea. Possible problems are briefly discussed below.

Sensitivity: When positioning with the help of an edge section, far less pixels are used than when computing the center of gravity. Therefore, the computation of the center of gravity is less sensitive except for drastic shape changes as in Fig. 3.5. If, for example, the entire object seems to shrink due to a slight overexposure because the sensor elements along the object edge get too much light, the center of gravity will hardly move thanks to the averaging over the entire object, whereas the edges move according to the apparent shrinkage of the object.

Fig. 3.8 Search line and detected upper contour (rendered prominent by a black line)

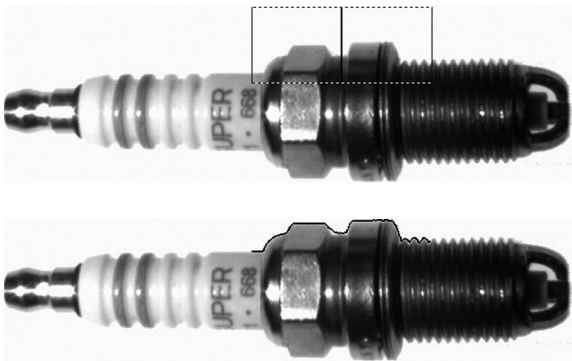


Fig. 3.9 Smoothing of a thread contour to create a smooth positioning edge



Edge quality: The edges used for position determination should be straight, smooth, as long as possible and always within the field of view so that they can be found reliably. Positioning using the small edge section in Fig. 3.8 was only possible because the object was fixed vertically, making it certain that the search line would cross the hexagon exactly at that point.

If there are any distortions along an edge, smoothing the edges vertically to the search direction may be helpful. Then, even irregular contours such as those of a thread can be used for positioning. Depending on whether the search line crosses the spark plug from the example on the lower edge of a winding or on the upper, there will be a difference in the y-coordinate of several pixels. However, if the thread is treated using a smoothing filter over a certain width, these differences can be eliminated by averaging, as is shown in Fig. 3.9. Even if the search line now moves within the smoothed area, it can still find the edge at the same y-coordinate.

Program hint: **NeuroCheck** contains a special filter function, *Smooth ROIs*, that was especially introduced for smoothing positioning edges.

Rotation: The biggest problem when positioning on an object edge are rotations of the object. Even if the object was only rotated around its center of gravity, its edges will move quite considerably, causing substantial pseudo-shifts. Furthermore, if one has to position on specific contour sections of complex shaped objects, the points in question might not be found at all. A rotation of the spark plug by five degrees leads to a movement of the center of gravity by 0.07 pixels (because of the spatial discretization at image capture, individual contour points are shifted out of place), the smoothed thread edge in Fig. 3.9 changes its position by an entire 20 pixels and is therefore completely useless for positioning.

Like the problem of edge quality, rotation can be managed with extended effort by determining the object's orientation and using it for position adjustment. Rotation adjustments are discussed in the following section.

3.2 Orientation of an Individual Object

Measuring the orientation of an object is usually much more difficult than determining its position. Symmetrical objects are especially difficult and an unambiguous determination is perhaps not even possible; but even objects whose

orientation is clear to a human can sometimes cause surprising problems, especially if one has to make a compromise between the exactness of the orientation determination and computation speed. A contour's center of gravity can be located relatively quickly and with high precision by averaging the coordinates of all contour points; determining an object's principal axis of inertia necessitates more effort, as will be seen later on.

3.2.1 Orientation Computation Using Principal Axis

In the following, some theoretical fundamentals for both the precise and approximate computation of the principal axis of inertia are explained. Modern image processing systems usually offer this functionality already built in, but there are some traps lurking that can be easier understood and avoided if one develops a sense of the underlying processes.

Formal computation of the principal axis of inertia: Every object has a principal axis, i.e. a line for which it has its smallest moment of inertia and around which it can therefore steadily rotate. According to Russ (2007), the angle of this axis with regard to the horizontal can be computed as follows:

$$\theta = \tan^{-1} \left\{ \frac{M_{xx} - M_{yy} + \sqrt{(M_{xx} - M_{yy})^2 + 4M_{xy}^2}}{2M_{xy}} \right\}. \quad (3.1)$$

where M_{ij} are the elements of the inertia tensor, which, using A as total area of the object, can be calculated as follows by summation over all points i of the object:

$$M_{xx} = \sum_i x_i^2 - \frac{(\sum_i x_i)^2}{A}, \quad (3.2)$$

$$M_{yy} = \sum_i y_i^2 - \frac{(\sum_i y_i)^2}{A}, \quad (3.3)$$

$$M_{xy} = \sum_i x_i y_i - \frac{\sum_i x_i \sum_i y_i}{A}. \quad (3.4)$$

So for each point of the object three multiplications and five additions have to be performed; for the spark plug serving as example object with an area of more than 60000 pixels, this results in more than 180000 multiplications and 300000 addition operations. In principle, it is of course possible to reduce the number of necessary operations by using only the contour points instead of the total area. This creates an error whose size depends on the shape of the object, because complex sections of the contour that include many indentations are overvalued.

Furthermore, a restriction to the contour results in an enhanced sensitivity of the computation of the axis to shape changes of the object, as can be seen in Fig. 3.5.

Approximate computation of the principal axis: Due to the high computation effort required to determine the principal axis with formal precision, approximation procedures are usually used. A possible approximation is given in Parker (1994). It is assumed that the principal axis always runs through the center of gravity and one contour point. Thereby, the number of candidate lines has been reduced considerably so that correspondingly fewer computations have to be performed. Other procedures use bounding polygons to determine the axis. This way, additional speed-up can be achieved, but a number of special cases have to be taken into account, e.g. singularities resulting from the degeneration of the polygon to a rectangle.

Problems using the principal axis: Computing the orientation using the principal axis causes certain problems regardless of whether one undertakes the computation effort for a mathematically exact formulation or employs approximation.

Orientation: The principal axis method can be used to compute the orientation of the object while the *direction* of the object, i.e. front and back, cannot be computed. To determine the direction of an object, it's helpful to use an auxiliary construction that takes into consideration the position of the object's center of gravity with regard to the middle of the axis. For *line-symmetrical* objects even small changes of contour can tilt the calculated direction by 180° .

Point symmetry: For objects that are approximately point-symmetrical, i.e. their shapes are similar to a circle or square, both principal moments of inertia differ so little that, instead of a relatively easily compensated tilt of the principal axis by 180° , the principal axis and secondary axis are switched. Because of this, the determined axis is tilted by 90° . The smaller the object, the greater the effect that changes on its contour will have. Figure 3.10 illustrates this using a simple object whose bounding rectangle is exactly square. In the case of the original object shown on the left, the (mathematically precise) computation of the principal axis leads to an angle of 0° according to the noted axis. Adding only one pixel in the upper left corner rotates the axis to 84° when using the entire area to calculate the axis, 80.9° when using the contour. If the above described approximation is used, this uncertainty always occurs for objects that have a height-to-width-ratio of 1, since in this case the bounding rectangle degenerates to a square.

Fig. 3.10 Effect of a minor contour change on the axis for approximately square objects

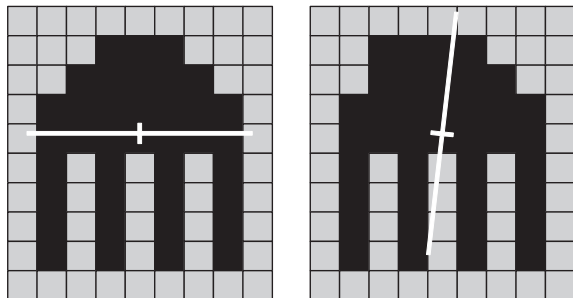
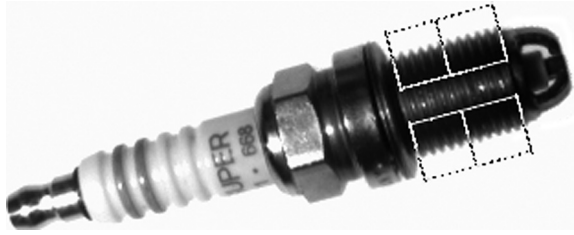


Fig. 3.11 Positioned search lines using principal axis approximation



Application of rotation adjustment: Now that we have said so many negative things about the principal axis computation, you may be wondering what to use the method for? The answer is that this method yields useful results for many objects—surprisingly good ones even if you adhere to some basic rules:

- The reference object should be long so that principal and secondary axis cannot be confused.
- If not only the direction but also the orientation of the object, i.e. the differentiation between front and rear, has to be determined, the reference object should be clearly asymmetrical relative to its secondary axis. Excellent results can be obtained by using, for example, elongated triangle shapes for which even approximations can yield results with precisions better than 0.5° .
- If only the direction has to be determined, straight edges are extremely well suited since, in this case, a straight line can be drawn between the starting and end point of the edge to determine the angle directly. Suppose, for example, that the spark plug has only been tilted slightly by $\pm 10^\circ$, the smoothed thread edge from Fig. 3.9 can then be used to determine the rotation. This way a precision better than 0.5° can be achieved.

Despite its long shape, the spark plug as a whole is not well-suited as a reference object for the approximation since it is almost ellipsoidal. One has to be prepared for errors in the range of 1.5° . But even this would be sufficient to reposition the search lines for the gauging task, as can be seen in Fig. 3.11. Since the thread contour is searched for separately after positioning and created as an object, the small angular deviation causes no errors worth mentioning. Of course one has to remember to re-orientate the measuring direction for the distance by the measured rotation angle.

3.2.2 Distance-Versus-Angle Signature

We saw in the last section that there are more or less quick and robust procedures to determine an object's principal axis of inertia and then to use this direction to compensate for rotations. But what can be done with objects that are notoriously ill-suited for corrections using the direction of the principal axis even if the

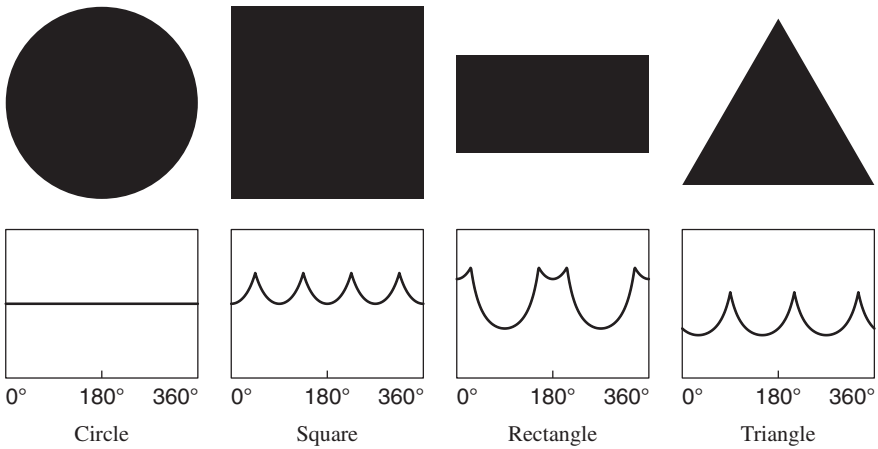


Fig. 3.12 Distance-versus-angle signatures for simple geometric figures

principal axis is computed mathematically correctly? One possibility is the *distance-versus-angle signature*.

Definition of the distance-versus-angle signature: The distance-versus-angle signature is defined according to Gonzalez and Woods (2008) as a one-dimensional representation of the two-dimensional figure *contour* based on the distance between the contour point and the object's center of gravity. The distance-versus-angle signature is thus a vector in which each element contains the distance between a contour point and the center of gravity in a certain direction. The number of elements is not fixed; it makes sense to measure the distance in steps of one degree, thus obtaining a projection consisting of 360 elements.

The distance-versus-angle signature of an ideal circle is a vector consisting of identical elements, each one the size of the circle's radius. Figure 3.12 shows distance-versus-angle signatures for some simple figures. The computation of the distance-versus-angle signatures is in principle quite simple; however, in practice, difficulties typical for the processing of real-world objects are encountered, which are not always trivial to solve. A half-line from the center of gravity to the contour can cross the contour several times if it is not an extremely simple geometric shape. In this case one has to decide which value to use. Usually, the distance to the point farthest away is chosen. Another problem occurs with relatively small objects for which there is not a contour point for every angle, making interpolation necessary.

Despite the conceptual simplicity, the computation effort for determining the distance-versus-angle signature is quite considerable. For each half-line, a straight line representation has to be created from the given angle.¹ All points of

¹ If one does not want to compute the coordinates of each point using trigonometric functions which would be extremely slow.

intersection of this line with the contour have to be determined to obtain 360 last points of intersection with a resolution of 1° . For each of these, the distance to the center of gravity has to be determined, which necessitates two subtractions, two multiplications, and one addition (taking the square root is not necessary since only the comparison is important, not the absolute values).

Use of distance-versus-angle signature: If the distance-versus-angle signature is available for the reference object and the current object, both have to be compared to determine the current orientation of the object. The basic procedure consists of overlaying the signature of the current object with the reference signature, calculating the similarity measure, moving the reference section to the next row, calculating the similarity measure again, etc., until the orientation is found for which the signatures are most similar. Different similarity measures can be used for this purpose, e.g. the Euclidean distance or a correlation coefficient, the latter having the desired or—depending on the application—undesired property of ignoring the actual dimensions of the objects.

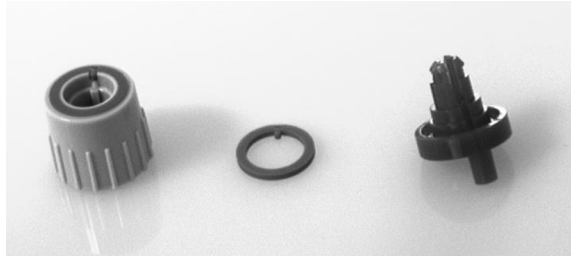
Graphically, one can imagine that the contour of the reference object is rotated in steps corresponding to the resolution of the signature over the current object until the contours “snap into place”. Both signatures have to be viewed as cyclically closed sequences, not as simple linear vectors. Using this procedure, orientation of difficult objects can also be determined with high accuracy. Even contour deformations as in Fig. 3.5 do not lessen accuracy because as long as the contour is not completely disfigured, the best similarity will still be determined for the same spot, only the absolute value of the similarity measure will decrease.

However, a word of warning is appropriate: The main advantage of the distance-versus-angle signature is its higher robustness for objects with very similar first and second moments of inertia and for disfigured contours. The precision of this procedure is necessarily limited to the signature’s resolution, i.e. for a resolution of 1° the orientation cannot be determined more accurately. This accuracy cannot be arbitrarily increased because when the resolution is increased to 0.1° not only is the computational effort considerably increased but also the demands on the reference objects, which have to have a contour of corresponding length to be able to fully exploit this resolution. Especially for gauging tasks with high demands on precision, no additional correction can substitute a careful build-up with precise and robust positioning of the objects.

3.3 Robot Positioning

The examples given so far might appear to be somewhat academic, especially seen in the light of our claim to show in this book how image processing can be used in industry. Thus, we will now make good on our promise to show real world applications that have been realized in manufacturing. We will also show the relation between rotation and translation adjustments and the use of entire object configurations as position reference.

Fig. 3.13 Control knob for electronic device with component parts



3.3.1 Application

Figure 3.13 shows the control knob of an electronic device and its component parts. These control knobs are automatically assembled. The relevant work step is the insertion of a rubber ring into the body, which proceeds as follows:

1. The rubber rings run on a line into a feeder.
2. The separated rings slip onto a turntable.
3. A gripper lifts the ring from the turntable and puts it into the body.

This process would be completely unproblematic if this were an evenly circular rubber ring, but, as you can see from Fig. 3.14, the ring exhibits an inward pointing “nose” and an opening so there is only one correct orientation to fit it into the control knob. Therefore, the ring has to be presented to the gripper in the correct orientation. To determine the ring’s orientation on the turntable after the feeder was, in this case, the task of the image processing system.

Figure 3.15 shows the structure schematically: The transparent turntable is illuminated from below so that the ring and its opening are clearly discernible as in Fig. 3.14. The turntable is captured by the camera; the image processing system determines the position and orientation of the ring and reports these values to the control system of the turntable and gripper. The turntable adjusts the ring in the predetermined orientation; the gripper moves to the corresponding position, picks up the ring, and fixes it in the intended position.²

3.3.2 Image Processing Components

The task at hand does not make any especially high demands with regard to optical quality. The rubber ring has an outside diameter of approx. 17 mm. The area of the turntable on which the ring can be found is about 40 mm so that for a camera

² The control system must of course take into account the angle by which the turntable has already been turned in accordance with the ring’s orientation, but this takes a lot less effort than capturing the image one more time and determining the position.



Fig. 3.14 Rubber ring for control knob in back lighting

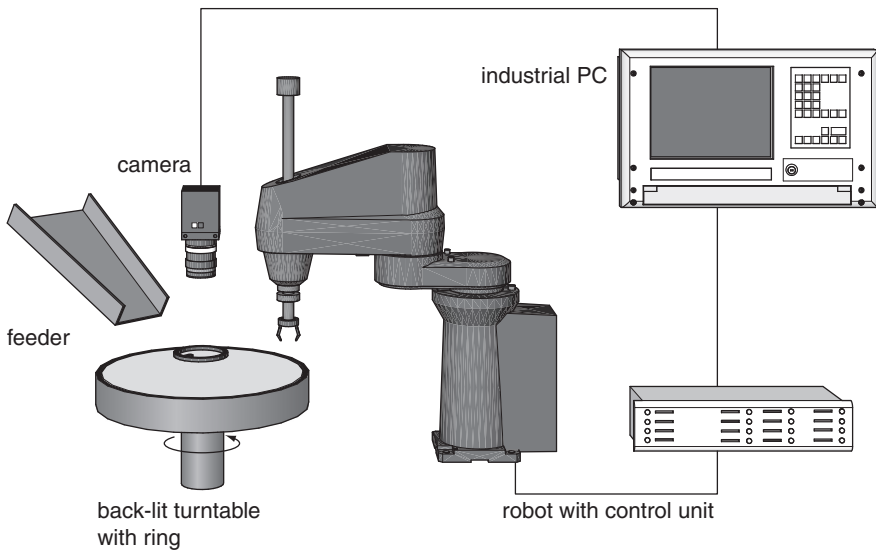


Fig. 3.15 Schematic structure of the positioning system for ring assembly

image size of 1032×776 , a pixel corresponds to about 0.05 mm. Even if the measurement is only accurate to two or three pixels, the gripper would have no difficulties in picking up the part.

The turntable consists of semitransparent plastic illuminated from below, creating a homogeneously illuminated background from which the ring is clearly set off.

Because of the negligible optical requirements, the selection of the lens is not critical for this task. It is mainly determined by the installation conditions, i.e. the distance that has to be maintained between the turntable and the camera to maintain the gripper's freedom of movement. The depth of field presents no problem because of the insignificant thickness of the ring, good illumination conditions, and the nature of the task.

3.3.3 Position Determination on One Object

Determining the ring's position on the turntable is extremely simple for such a clearly defined object. To find the ring as an object is no problem because of the superb contrast of the back-lit image, which makes it very easy to compute the center of gravity. But this solves only half of the task since the orientation is still unknown. Of course the orientation cannot be determined using the circular outer contour. After what has been said so far, the obvious choice would be to use the inner contour, which holds a clearly defined reference point in the form of the pronounced “nose” to orient the part.

Using the distance-versus-angle signature, it is quite simple to determine the ring's orientation. The principal axis method does not work robustly for the known reasons since the contour is almost circular. Figure 3.16 shows the result of the position and rotation determination using a tracking search line. The leftmost image shows the reference position and orientation. As you can see, the defined search line crosses the nose and the opening in the ring.

Program hint: The inner contour and the ring opening are easy to find according to our approach by starting a second search for bright objects within the outer contour. If the object search is limited to the largest object, only the inner contour is found. You'll find this example in the download in the directory Examples\Positioning.

3.3.4 Orientation of an Object Group

There is another way to determine the orientation with a high degree of security and precision. From the point of view of an image processing system there are two light objects within the outer contour of the ring, although a human might not think this way: the “inner space” of the ring with its characteristic nose, which is used to determine the orientation, and the neighboring small opening. The line from the center point of the entire ring to the center of gravity of this opening gives a reference to orient the ring that is just as good as the distance-versus-angle signature from the previous section. Figure 3.17 shows the two objects; the background is colored black to tell it apart from the ring.

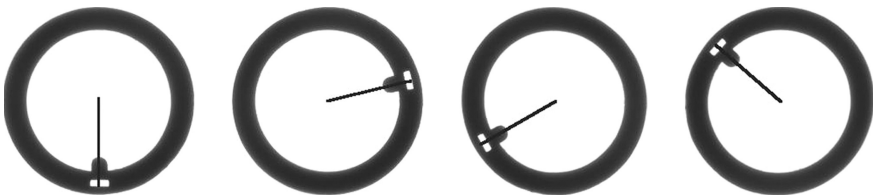


Fig. 3.16 Result of re-positioning using the distance-versus-angle signature of the inner contour

Fig. 3.17 Object group for the determination of the rotation



After both objects have been found, the direction of the connecting line is easy to determine by inserting both centers of gravity into a linear equation and computing the angle as arc tangent of the slope. However, this result is not unambiguous since the arc tangent has a period of 180° . We still need the orientation. It can be easily determined by defining that the connecting line always has to start at the bigger of the two objects, i.e. the interior of the ring. From the object search, the size of both objects is known, so the defined direction can be easily ascertained. The quality of the determination is just as good as using the distance-versus-angle signature method for the inner contour, which was used above, but calculation is by far less computation-intensive so significant time saving is possible for large objects.

Program hint: If the **Group of regions** option is used together with the “ $\pm 180^\circ$ ” selection in the *Determine Position of ROIs* function when determining the reference position, **NeuroCheck** proceeds exactly in the manner described.

In summary, the determination of position and orientation using the arrangement of several objects can be accomplished reliably, robustly, and quickly. In some cases it is the only possibility, e.g. for rotationally symmetrical parts whose orientations can only be determined using surface characteristics.

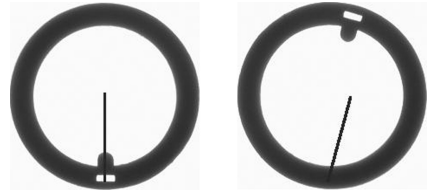
3.3.5 Comments Concerning Position Adjustment

As could be seen above, there are a number of possibilities for determining position and orientation that can be combined in different ways. Attention must be paid to some factors that can lead to subtle errors that cannot always be detected at once.

Pivot: It is obvious that it does not suffice to shift and rotate ROIs in place whose position and orientation are to be adjusted. They also have to be rotated around their pivot point. What happens otherwise can be seen in Fig. 3.18. There, the search lines were shifted according to the translation measured but were only rotated in place. One can see that they conform to the ring’s new orientation, but they are in a totally wrong location.

This error is, of course, quite obvious and will therefore hardly ever occur. A more subtle error can occur when translation and rotation are measured using different objects. This case occurs quite frequently because, as could be seen in Fig. 3.9, object edges are extremely well suited for rotation measurements. But the

Fig. 3.18 Missing rotation around the pivot



translation cannot be simultaneously measured on the same edge since the edges do not rotate around themselves but around the center of gravity of the entire object. The edge's center of gravity exhibits pseudo-shift. In a case such as this, translation will be measured using the entire object, rotation using the edge. Rotation has to be executed around the reference object of the translation, as can be seen in Fig. 3.19. Rotation measurement is performed, as in Fig. 3.9, on the smoothed thread edges. In the top image, rotation was performed correctly around the entire object's center of gravity; in the bottom image, it was incorrectly performed around the reference edge's center of gravity.

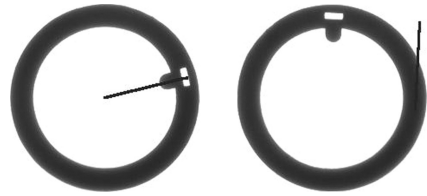
Sequence of rotation and shift: Normally, you only need to pay attention to this aspect if you program image processing applications yourself, but even working with ready-to-use image processing software can cause situations in which you have to pay attention to the sequence of rotation and shift adjustment. This can apply when subsequent adjustments are necessary to be able to find different reference objects. In this case, translation adjustment has to take place *before* rotation adjustment. Figure 3.20 shows the effect occurring when, in the second image of Fig. 3.16, first the line's orientation is adjusted and then the shift is performed.

Pseudo-shifts: It has already been mentioned that shape changes can lead to apparent shifts of the center of gravity. This effect can also occur without an optically induced shape change. Because of the nose, the center of gravity of the ring's interior that was used in Sect. 3.3.3 to determine the rotation is not precisely in the ring's center. If the ring is rotated, the interior's center of gravity shifts in relation to the outer contour's center of gravity. This does not cause any problems for this inspection task which has relatively low precision requirements; however, for more finite task settings, these effects have to be taken into account.

Fig. 3.19 Rotation around a wrong pivot



Fig. 3.20 Separation of translation and rotation adjustment, on the left in the correct, on the right in the wrong sequence



It is advantageous for some applications not to measure the shift using the object's center of gravity but using the vertical or horizontal dimensions, e.g. the vertical shift using the uppermost point. This is only possible if one makes sure that the object cannot rotate, for a rotation around the center of gravity causes a shift of the outermost points of the object unless it is circular. In principle, it is possible to calculate this pseudo-shift in reverse if the rotation is known, but for complicated object contours this can require considerable effort.

3.4 Summary

Importance of position determinations: They are a part of most image processing applications since position and orientation of parts to be checked in front of the camera can only rarely be reproduced with absolute precision. In such a case, the position determination is only an intermediate step, a tool for executing the inspection task itself.

Robot positioning: In some cases, the determination of the position is the real goal of the image processing application. A typical example for this is robot positioning, where the position of the object to be checked is given to a robot that is to grip the object.

Translation adjustment: The translation adjustment determines and adjusts the test piece's shift in relation to a reference position. Position determination is usually done using the center of gravity, but the object's edges, i.e. the outermost points in one direction, can also be used. Shape changes, for example due to illumination effects and object rotation can cause pseudo-shifts.

Rotation adjustment: For a rotation adjustment, the test piece's rotation in relation to a reference position is determined and adjusted. Commonly used methods for the determination of the rotation are the principal axis computation and the distance-versus-angle signature.

Principal axis method: A formal exact computation of the principal axis is very computation-intensive, approximations are not suited for all objects. Especially objects with a height-to-width-ratio close to 1 are problematic since the axis can suddenly tilt by 90°. Long objects such as edges, possibly after prior smoothing, are ideally suited. Furthermore, principal axis computations are strongly influenced by shape changes, for example due to illumination changes.

Distance-versus-angle signature: This method for the determination of rotation measures the distances between the object's center of gravity and the object's contour in certain angle increments (typically one degree) and inserts these into a projection vector. This vector is then correlated to the projection vector of the object in reference position. Usually, this method is not as computation-intensive as a mathematically exact principal axis computation, but far more robust than approximations. The accuracy depends on the angle increment used and cannot be increased arbitrarily for small objects with few contour points.

Hierarchical adjustments: Position adjustments can be hierarchically sequenced. This is frequently the case in complex image scenes, in which the reference object itself can only be found after a position adjustment.

Object groups as reference: Object groups represent a very robust reference for position and rotation determination, which can also be used for point symmetrical objects with identifiable surface features. Furthermore, this method is very fast since only the centers of gravity of the segmented objects are required (and, if need be, the sizes of the objects when a 180° orientation has to be determined).

Rotation determination using edges: An edge used for the determination of a rotation should not be used to compute the translation since a rotation around the entire object's center of gravity causes a pseudo-shift of the edge.

Translation determination using edges: If the translation is determined using edges instead of centers of gravity, one has to make sure that no pseudo-shift can occur because of the object's rotation.

References

- Gonzalez RC, Woods RE (2008) Digital image processing, 3rd edn. Pearson Education, Upper Saddle River
Parker JR (1994) Practical computer vision using C. Wiley, New York
Russ JC (2007) The image processing handbook, 5th edn. CRC Press, Boca Raton

Chapter 4

Overview: Segmentation

The concept of an *object* is central to the solution approach in Sect. 1.5. Indeed, it is decisive for the nature of industrial image processing since its purpose is always to gather information about objects existing in the real world represented in image scenes. In the introductory example in Sect. 1.6 and throughout Chap. 3 on positioning we have frequently used *segmentation* methods, i.e. algorithms which isolate objects from the scene. In these sections we have simply assumed that such methods exist and achieve the desired effects. Over time a number of such methods have been developed. The most important and most commonly used will be introduced in this chapter.

4.1 Regions of Interest (ROIs)

We will start this section with a chapter on *regions of interest (ROIs)*. We have already mentioned in Sect. 1.5 that our approach to image processing applications does not make a fundamental distinction between objects segmented in the image scene and those manually defined by the user. This equivalence is unusual in image processing and therefore deserves some explanation before we start to deal with actual image objects. Besides, some aspects of the object concept are more easily explained using manually defined objects, since their behavior is less complex than that of segmented objects.

Applications without segmentation: It is quite possible to do completely without segmentation in an image processing application. For example, to inspect the surface of a work piece, it may be sufficient to compute some characteristic features from a predefined area of the image (or the entire image), like the average brightness, the variation of the gray values, the occurrence of gray level transitions, etc.

On a surface supposed to be uniformly white, the existence of scratches or pollution can simply be determined by the existence of high contrast or strong gray level gradients. On the other hand, the inspection task can also require such features to be computed for the surface area of an object, which has to be isolated

in the image scene first. If, however, the same feature computation is to be performed on two different structures, it appears sensible—at least from a programmer’s point of view—to generalize in such a way that both structures, the manually defined region of interest and the automatically found image object, can be treated identically. This gives an initial motivation for an equivalence of these two structures.

Hierarchical search: In the preceding examples in Sect. 1.6 and Chap. 3, we approached the segmentation task by defining rectangles or search lines inside of which (in the case of rectangles) or along which (in the case of search lines) individual digits or object contours were detected. This is the typical method of using regions of interests (search regions, test windows, areas of interest, ROIs) in image processing: as a limitation of the area within which image objects are to be segmented.

Hierarchical structures are quite common in engineering: on an engine, a type plate is mounted on which characters have to be read; components are grouped on a circuit board; on the surface of a casing, bore holes can be seen which in turn contain threads; the list is endless and suggests emulating this hierarchical structure when searching for image objects. Instead of scanning the entire engine for characters, the plate is searched for first, since it is much simpler to find; then the characters are detected. This means that on the image object “plate”, which has already been found through segmentation, exactly the same operation is performed as inside a manually defined search area. So this again indicates a strong reason to treat both image objects and regions of interest in the same manner.

Differences between regions of interest and objects: It is obvious that manually defined regions of interest and objects found by segmentation will behave differently in certain respects. For example, it is perfectly possible to determine the position of a manually defined region of interest, but it is also useless because it will always be at the same place and in the same orientation. Conversely, a position adjustment can be performed on segmented objects but is typically not very useful, as they already are at their true positions in the image.

4.2 Binary Segmentation

Several times in Chap. 3 we have referred to detecting a transition between light and dark inside a search area or on a search line. Methods based on distinguishing light and dark areas are called thresholding methods because the two states, typically denoted as Black and White, are usually distinguished by a threshold. The following factors are decisive for segmentation by thresholding:

- How Black and White are created: we start with a gray level image with 256 levels of brightness so we have to determine what brightness areas of the gray level image are depicted in the binary states white and black. The most

frequently used method, which we will discuss in detail, uses the brightness of the pixels directly and defines a single value as the threshold.

- How the object representations are created: Basically we can distinguish between contour-based and region-based methods. Contour-based methods search for the outline of an object first, then derive the set of points making up the object from the contour. Region-based methods work outward from a point of the desired color until a contiguous region of uniform color (black or white) has been created.

4.2.1 Thresholds

This section deals with the first aspect of thresholding methods: how to derive the threshold for distinguishing between the areas in the gray level image which will appear black or white in the bi-level image.

Global thresholds: The simplest method of transforming a gray level image into a bi-level image uses a single gray level for the whole image. All gray values above that threshold are defined as white, all values below as black. This value is referred to as the *threshold*.

Figure 4.1 shows the effect of a global threshold on a gray level image. It can be clearly seen that the distinction between the bright metal contacts and the dark plastic does not work equally well in all areas of the image. Due to varying angles of incidence, the brightness levels of the contacts are too different to use a global threshold. That's why the left object is breaking up.

Location-dependent thresholds: Problems with segmentation using global thresholds will always occur when brightness properties of relevant image areas differ. Global thresholding does not allow for the segmentation of images with uneven lighting. It is also unsuitable for images where the contrast between objects and background is different in different areas of the image. Figure 4.2 demonstrates that the object from Fig. 4.1 will not break up when local thresholds are used. For every region of interest an individual threshold was computed.

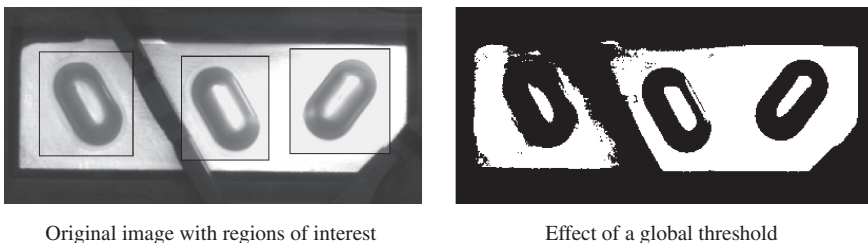


Fig. 4.1 Thresholding a *gray* level image using a global threshold

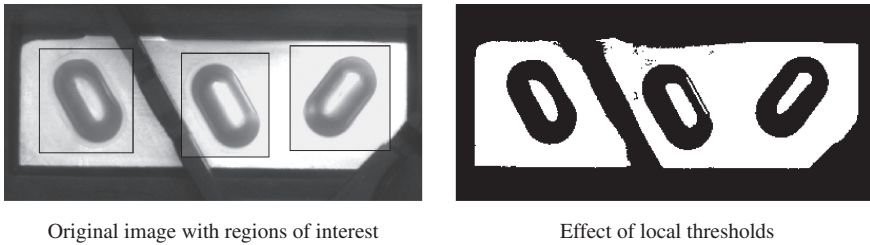


Fig. 4.2 Thresholding a *gray* level image using local thresholds

In this case, segmentation is easily accomplished by allowing for different thresholds in different areas of the image. The consistent application of image objects or regions of interest in our approach from Sect. 1.5 naturally supports such a differentiated procedure by assigning an individual threshold as a property of each region of interest. The segmentation function then uses the respective property for segmentation in each region of interest.

Note that for the methods to determine thresholds discussed so far, it is not necessary to actually create a bi-level image for the segmentation process. Instead it is sufficient to decide for every pixel whether its gray level is above or below the threshold to assign it to a segmented object.

It would not be practical, however, to use a local threshold for every single pixel in the source image to give an extreme example. Sometimes such a costly method is appropriate for images of poor quality; often, though, it will be preferable to apply a preprocessing operation to the image, after which it can be segmented using a smaller number of local thresholds (preprocessing methods are discussed in Chap. 2).

4.2.2 Threshold Determination from Histogram Analyses

So far we have assumed that the threshold (or the thresholds, when local ones are used) is simply predefined and indeed there are applications where one can set a certain fixed gray level as the threshold. Especially under production conditions image properties will frequently not be sufficiently stable to proceed in this way. Illumination can vary and, more importantly, the appearance of objects can change. Test pieces can be soiled, contrast properties can vary between parts, as Fig. 4.3 shows. In each of the two images, segmentation can be performed with a global threshold, but the same threshold cannot be used for both images. Typically, one will have to adapt the threshold to the properties of the current image—or every single region of interest in the current image for local thresholds.

Program hint: **NeuroCheck** supports the use of local thresholds through the option to optimize the threshold for each group of regions separately using bimodality analysis (see Sect. 4.2.3).



Fig. 4.3 Segmentation with fixed threshold and various contrast properties

A typical method for determining thresholds from image properties is to analyze the gray level histogram of the image—or individual image areas. The following section will therefore discuss gray level histograms and analysis methods.

4.2.3 Gray Level Histograms

Binary thresholds as a means for segmentation are based on the brightness properties of image areas. *Gray level histograms* are the typical tool for the representation of brightness characteristics.

Definition: In descriptive statistics, a histogram denotes the representation of a frequency table in a bar chart. We will use it here for the bar chart as well as for the underlying frequency table. The frequency table records how often values from specified value ranges occur. The same procedure is used for gray level histograms. For each gray level value the number of occurrences in the entire image or part thereof is recorded in the histogram.

Typically, these occurrences are counted individually for every single gray value. It is of course also possible to group several values into gray level classes, e.g. in steps of 16 gray levels. This would result in a histogram containing only 16 values instead of 256, the first value representing the frequency of gray levels between 0 and 15 in the image, the second value that of gray levels between 16 and 31, etc. We will always use histograms with the full resolution of 256 gray levels here. Computation of such a histogram for the entire image is very simple: first an array of 256 counters is allocated, then a loop is performed over all pixels in the image; for each pixel the array cell is incremented, which corresponds to the gray

level of the pixel. Computing a histogram for individual image areas of possibly arbitrary shape, i.e. segmented objects, is somewhat more complex, since we have to take care to count only pixels contained in the respective region. A very important aspect still holds, though: only the gray level of a single pixel is used in each step. Therefore, the histogram is a first-order statistical description for the distribution of gray levels in an image (region). Examples for second-order statistics can be found in the field of texture analysis where spatial relationships between gray levels are examined.

Test for bimodality: Sahoo et al. (1988) give an extensive overview of methods for computing thresholds from gray level histograms. We will restrict ourselves here to the bimodality test discussed in detail in (Sonka et al. 2008). The method is based on the assumption that a typical histogram shows two prominent peaks, one for the background brightness, one for the objects. The histogram of the image of a relay point in Fig. 4.4 shows that industrial image scenes often meet that assumption very well. One can see that the prerequisites for bimodality are well met, which is also true for many industrial images.

With this method, the threshold is typically placed at the minimum between the two most prominent peaks. Frequently, special measures will have to be taken to avoid this threshold being situated between two local maxima belonging to the same global maximum. This can be done for example by postulating a minimum distance between the two maxima used for setting the threshold or by smoothing the histogram prior to computing the threshold.

Despite the obvious simplicity of the approach, many refinements of bimodality testing have been developed and investigated, only a few of which we can mention here. More methods are discussed in (Sonka et al. 2008) and (Gonzalez and Woods 2008). There you will also find an exhaustive list of literature on the subject.

Empirically, superior segmentation results can be obtained by shifting the threshold according to the area of the two main maximum leads. The threshold is then placed at the common center of gravity of the two peaks. This method can lead to a problem, though, if one of the maxima is strongly predominant. In this case, the threshold will move into the area of the predominant peak. A non-linear scaling of the histogram can avoid this effect, e.g. by using a square-root or logarithmic transformation. Figure 4.5 demonstrates the effect of these scaling methods on the shape of the histogram and the segmentation result.

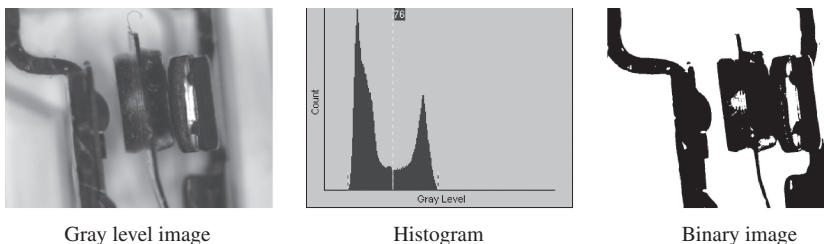


Fig. 4.4 Threshold computation by histogram shape analysis

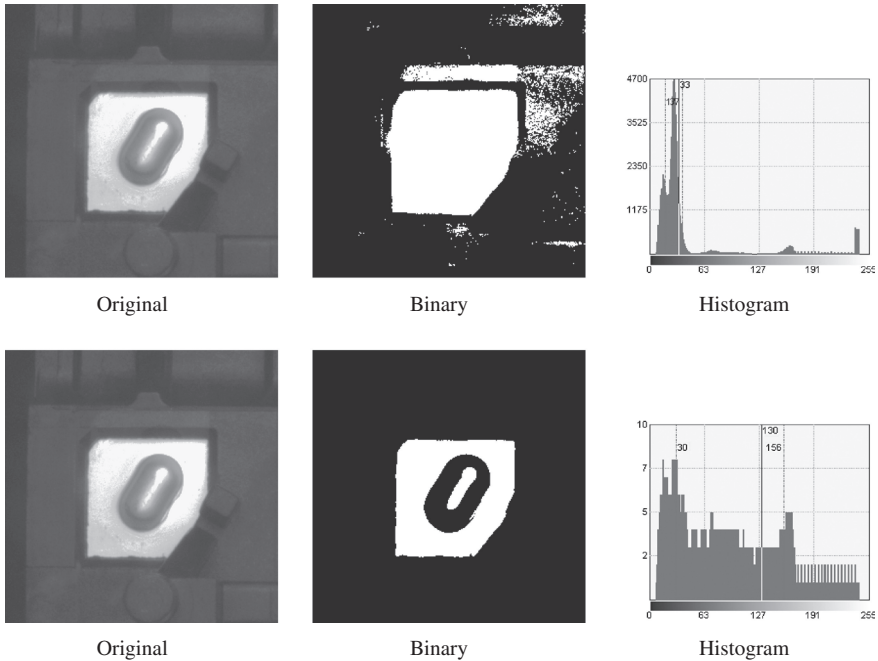


Fig. 4.5 Non-linear transformation of an uneven histogram

Artifacts are a frequent problem in practical applications. Artifacts denote image areas found as objects through segmentation although they do not correspond to actual objects in the real-world scene. For example, these can be reflections segmented as bright objects or shadows found as dark ones. Usually, such artifacts can be excluded from segmentation by prior knowledge or because of their size, but they pose a problem for histogram analysis. Such artifacts are typically restricted to narrow ranges of extremely low or high gray levels, i.e. they are either very bright or very dark. Thus they form prominent maxima exerting a strong influence on the computation of thresholds by bimodality testing although they are actually meaningless as far as the analysis of the scene is concerned. In such cases it is advantageous to ignore the outer ranges of the histogram.

Program hint: For the transformation of uneven histograms, the check function *Determine Threshold* in **NeuroCheck** offers the “Histogram equalization” option, which scales a histogram by the square-root or the logarithmic function. Furthermore, regions of interest containing no objects at all can be excluded from the computation of the histogram using the “Minimum contrast” parameter. This prevents uniform background areas from affecting the threshold computation. Finally, artifacts can be excluded from the computation by the “Defect suppression” parameter.

Of course there are a number of other approaches to threshold determination. For example, all pixels from areas with high gradients can be excluded. Since

strong gray level gradients occur at the location of object edges, the histogram will contain only pixels located inside objects or inside the background, not on edges. The minimum of the histogram will then be more pronounced and thus easier to find. It is also possible to work the other way around and use only pixels located at edges (consequently placing the threshold at the maximum of the histogram since the histogram contains only pixels from areas separating objects and background).

4.2.4 Generalizations of Thresholding

Range thresholding: Up to now we have discussed only images in which the background was bright, the relevant objects dark or vice versa. It is, of course, also possible that the objects are of medium brightness and the background consists of dark and light areas. Examples of such a segmentation problem include sealing lips or a glue thread that separate a bright inside area and a dark background.

The reverse situation is also possible. For example, the above-mentioned thresholding methods are unable to separate a checkerboard pattern from a uniform gray background. In manufacturing, such an effect can occur for stamped digits. Depending on the angle of incidence of the light, parts of the edges of a stamped digit may appear bright, other parts dark, whereas the mostly planar background shows a uniform appearance.

There is a number of “tricks” to remedy such a situation. The look-up tables from Sect. 2.1.1 are an example of a preprocessing method enabling segmentation of such image material using a simple threshold. As useful as look-up tables may be, however, they have some disadvantages in such a situation:

- Although look-up tables are one of the fastest methods in preprocessing, they still require transformation of every single pixel, at least within the relevant regions of interest.
- Standard look-up tables are not context sensitive, i.e. they process the entire image in the same way. It is of course possible to adapt a look-up table to the neighborhood of the current image pixel, but this leads to a considerable increase in structural and computational complexity.
- Static look-up tables are difficult to adjust to the requirements of such situations; a lot of visual control and iterative work will be required to create a look-up table which will lead to satisfactory results under the typical variations of industrial image scenes.

A much more elegant and simple method uses the concepts of *logically white* and *logically black*. Normal thresholding as we have discussed above defines all image pixels above a certain threshold as white, all others as black. Of course, it is perfectly possible to introduce another higher threshold and define all pixels with a brightness above this threshold again as black. This is, of course, contrary to visual perception, but allows for isolating objects or background areas of medium

brightness without particular additional effort compared to “normal” thresholding. Figure 4.6 demonstrates the principle of range thresholding. In this case, the range between gray levels 100 and 150 has been defined as logically white.

All the histogram analysis tools can be used for the determination of these thresholds, although complexity, of course, increases due to possible interdependencies between the thresholds. Further generalization leads to multi-thresholding techniques allowing for the segmentation of objects within several gray level ranges in one step.

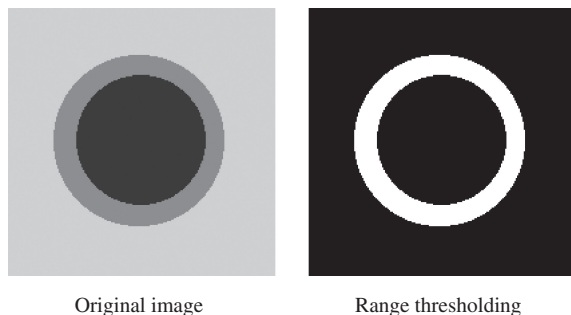
Thresholds not based on brightness: Finally we would like to mention that the transformation of a gray level image into a binary image does not necessarily have to be based on brightness thresholds. Generally speaking, the distinction between (logical) white and (logical) black pixels is a kind of classification and nowhere is it written that this classification must use “brightness” as the only feature.

For example, a very simple generalization would be to decide between black and white according to the average brightness in a neighborhood of each pixel. This would compensate for small image defects. This generalization and many others can also be achieved by a standard thresholding combined with a previous filtering of the image, e.g. using a mean value filter.

More complex methods based on actual pixel classification compute various features for the neighborhood of every pixel, from simple features like average brightness to more complicated ones like second-order gray level statistics describing the spatial distribution of pixels, which humans perceive as *texture*. Depending on the classification method, complex relationships between these features can be taken into consideration. One has to keep in mind, however, that texture features are particularly computation-intensive and hardly suited for real-time segmentation.

The combination of preprocessing methods and pixel classification with standard thresholding allows for very complex segmentation processes. The resulting binary image, whether it be created by a simple brightness thresholding or a complex classification of pixel features, is geometrically identical to the original gray level image. The description of an object or region of interests segmented from the binary image therefore allows this object to be analyzed either in the

Fig. 4.6 Principle of range thresholding



binary image or in the original image. An introduction to using multiple criteria for thresholding can be found in (Russ 2007).

4.3 Contour Tracing

In the previous section we have discussed the first aspect of binary segmentation, the distinction between (logically) bright and dark image areas. Now we turn to the second aspect, the creation of the object representation. A common method which has been used in the preceding examples uses *contour tracing*. The contour of an object is commonly understood as a closed line running along the border of the object, i.e. consisting exclusively of border pixels. This definition seems self-evident, but it is not quite as simple as it reads, because we first have to explain what exactly a border pixel is. This leads us to the concept of *connectedness*.

4.3.1 Connectedness

There are two common methods to define the neighborhood of a given image pixel in digital image processing: four- and eight-connectedness, as depicted in Fig. 4.7. The four-connectedness allows only vertical and horizontal movements between adjacent pixels, whereas the eight-connectedness also uses diagonal connections.

The concepts of pixel neighborhood and contour, which are immediately obvious to humans, are far from trivial for a computer. They are a good example for the importance of discretization in digital image processing. In the following, we will demonstrate how discretization in combination with different notions of connectedness can cause digital image processing systems to behave in striking contradiction to human perception. This behavior can confuse even experienced users—and sometimes pose unforeseen problems for programmers.

The collection of black dots in Fig. 4.8 running in 45° direction appears as a line to the human eye. However, when using four-connectedness, this arrangement

Fig. 4.7 Different definitions of pixel neighborhood

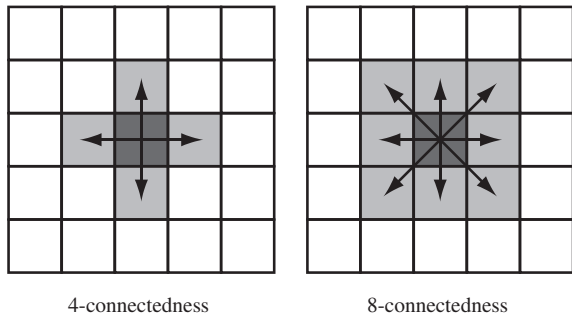
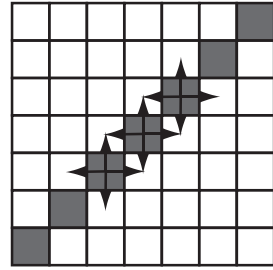


Fig. 4.8 Diagonal, unconnected in four-connectedness



of dots is not contiguous at all, since none of the black pixels is vertically or horizontally adjacent to another one. It seems reasonable to approximate the human perception of contiguous lines by using eight-connectedness. This, however, leads to other problems, as can be seen in Fig. 4.9. The contour of the black object in this figure is completely closed only when eight-connectedness is used. But in this case, the background is also contiguous, since the white pixels are also connected diagonally through the dark border. Background and object cut into each other, the background practically flows through the object contour. A hierarchical search for a light object inside a dark object can therefore lead to considerable demarcation problems.

Finally, it should be mentioned that even the notion of intersecting lines loses its obviousness in the discrete space of digital image processing. The mathematical definition of the point of intersection as the only common point of two lines is unfortunately not applicable in spatially discrete images. Figure 4.10 demonstrates that two lines defined using eight-connectedness can obviously intersect without having a single common pixel.

4.3.2 Generating Object Contours

The principle of generating objects by contour tracing is quite simple; an appropriate algorithm can be found, for example, in (Parker 1994):

Fig. 4.9 Intersection of object and background using eight-connectedness

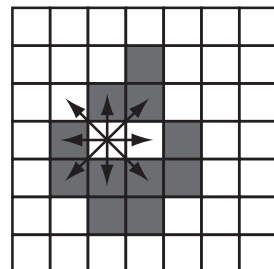
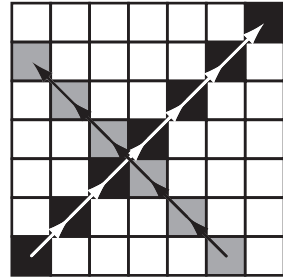


Fig. 4.10 Lines intersecting without a common point



1. The region of interest is searched for a first transition between background and object, i.e. a transition from (logical) white to black if dark objects are to be segmented, black to white for light objects.
2. As soon as the transition has been found, the algorithm searches for the next neighbor of the same (logical) color—with respect to the connectedness definition used. In this case, only border pixels are accepted as neighbors, i.e. pixels adjacent to at least one pixel of the other (logical) color in order to prevent the algorithm from swerving off into the interior of the object.
3. Next, the entire contour is traced in this manner, either clockwise or counter-clockwise (the direction does not matter as long as it is consistent).
4. Contour tracing ends when the algorithm reaches its starting point, i.e. the pixel where the first transition had been found.

The main advantage of this method compared to many others is that it always generates visually contiguous objects with closed border lines. As always, there are some additional considerations in real life, for example:

- Pixels touching the image border have to be counted as border pixels, or else no contours could be found along the image border.
- The same holds for pixels at the border of a search area if the search is to be restricted to this region.
- If contour tracing is performed not only in geometrically simple regions like rectangles but also—due to the equivalence of region and object in our approach—inside objects which have themselves been created by contour tracing, one has to take care that the paradox demonstrated in Fig. 4.9 is avoided.
- One must prevent the tracing algorithm from running back into itself at isolated points protruding from the object.

This list is far from exhaustive, of course; it is only meant to hint at what difficulties special cases can present to programmers and users alike.

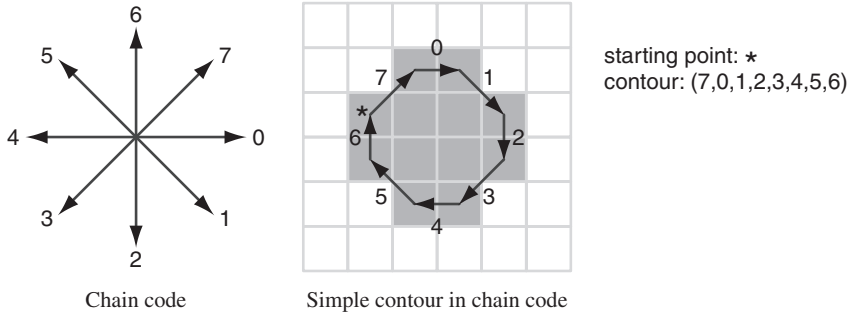


Fig. 4.11 Chain code for contour description

4.3.3 Contour Representation

Contours generated using the above algorithm can be represented as completely different data structures. Of these, the chain code has been generally accepted in digital image processing because it offers a number of algorithmic advantages. It is for example very simple to compute the perimeter and dimensions of an object from the chain code, also curvature and first derivative of the contour. It also facilitates smoothing the contour to lessen the influence of image noise.

The chain code, also called Freeman code (Freeman 1961), represents the contour as a sequence of directional information. The position of the starting point is stored separately. Figure 4.11 shows how the direction of the connection between two contour points is encoded as a single number.¹

4.4 Template Matching

All segmentation methods we have encountered so far in this chapter have one thing in common: objects are constructed “bottom-up” by performing certain transformations on the iconic level, like thresholding, edge detection etc., followed by a decision about the location of objects based on the transformed image. These methods therefore try to determine whether candidates for objects *of any kind* are present. This leads to the question if it would not be possible to search for specific objects in the first place thus saving the effort to look for the relevant objects from all possible objects. This is indeed possible, but it is not simple either. After all, you cannot tell a computer to look for a spark plug or certain characters in an

¹ In principle the assignment of numbers to directions is arbitrary. The one depicted here is a common convention, used e.g. in (Parker 1994), (Nischwitz et al. 2007), (Gonzalez and Woods 2008) and (Sonka et al. 2008).

image, for it is ignorant of what those objects are and how to identify them within an image.

This section will firstly present a common method for detecting *previously known* objects in an image, the template matching algorithm. At the end of this section we will discuss another method that also looks for previously known objects in an image.

4.4.1 Basic Operation

In a basic form, match-based segmentation works by moving an image of the object to be detected across the current image and computing a similarity measure at each position. If one position yields a significant similarity value the algorithm concludes that an instance of the respective object is present at that position. This procedure is easily generalized to an arbitrary number of object patterns to be detected and an arbitrary number of positions—thereby also increasing the required computation effort arbitrarily. We will come to optimization in a minute.

The *image of the object* or *pattern* is commonly denoted as a *template*, hence the term *template matching*. Let us now go into some more detail.

Similarity measure: Matching can be based on very different similarity measures, but the most common one by far is the correlation coefficient (Russ 2007; Sonka et al. 2008).

The empirical correlation coefficient of a series of number pairs $(g_1, t_1), (g_2, t_2), \dots, (g_N, t_N)$ is, according to Bronstein et al. (2005), defined by Eq. (4.1) where $\bar{g}$ and $\bar{t}$ are the mean values of the components g_i and t_i , σ_g and σ_t their respective empirical standard deviations.²

$$r(g, t) = \frac{\sum_{i=1}^N (g_i - \bar{g})(t_i - \bar{t})}{(N - 1)\sigma_g\sigma_t} . \quad (4.1)$$

The correlation of an image section with a template of the same size can be computed using this formula by writing the gray levels of the template and the image section as a vector of number pairs as shown above. Direct computation from the image may look a little more complicated at first, but hopefully Fig. 4.12 provides some clarification. A template t of width w and height h is positioned with its top left corner at position $(x, y) = (1; 1)$ of the image. The gray level $t_{i,j}$ of the template corresponds to the gray level $g_{x+i,y+j}$ of the image. The indices run from 0 to $w - 1$ for the x-coordinate, from 0 to $h - 1$ for the y-coordinate. The (implied) vector of number pairs is therefore $(g_{x,y}, t_{0,0}), (g_{x+1,y}, t_{1,0}), \dots, (g_{x,y+1}, t_{0,1}), \dots, (g_{x+b-1,y+h-1}, t_{b-1,h-1})$. In Fig. 4.12 only the correspondences on the four corner points are shown to avoid obscuring the image with too many lines.

² We use σ_g here because it is commonly associated with the standard deviation; strictly speaking it should be s .

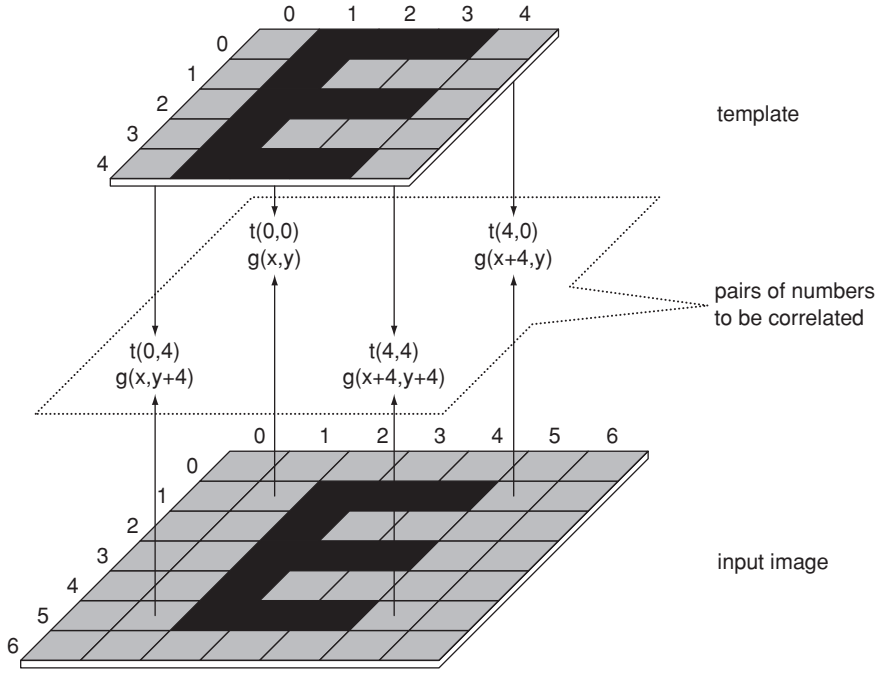


Fig. 4.12 Principle of template matching by correlation

The sums for the computation of the correlation coefficient have to be taken over both coordinates. With $N = w * h$ as the number of points in the template the correlation coefficient takes the form of Eq. (4.2).

$$r(x, y) = \frac{\sum_{i=0}^{b-1} \sum_{j=0}^{h-1} (g_{x+i,y+j} - \bar{g})(t_{ij} - \bar{t})}{(N-1)\sigma_g\sigma_t} . \quad (4.2)$$

The empirical mean values and standard deviations are computed as follows:

$$\begin{aligned} \bar{g} &= \frac{1}{N} \sum_{i=0}^{b-1} \sum_{j=0}^{h-1} g_{x+i,y+j}, \\ \bar{t} &= \frac{1}{N} \sum_{i=0}^{b-1} \sum_{j=0}^{h-1} t_{ij}, \\ \sigma_g &= \sqrt{\frac{1}{N-1} \left(\sum_{i=0}^{b-1} \sum_{j=0}^{h-1} (g_{x+i,y+j} - \bar{g})^2 \right)}, \\ \sigma_t &= \sqrt{\frac{1}{N-1} \left(\sum_{i=0}^{b-1} \sum_{j=0}^{h-1} (t_{ij} - \bar{t})^2 \right)}. \end{aligned}$$

As can be seen from the above equations, calculating the correlation between image section and template is basically a simple and systematic procedure, but requires quite some computation effort—in particular when we take into account that the most straightforward form of template matching shifts the template pixel by pixel across the entire image. This means that the above computation has to be performed for every single pixel (i.e. almost every one; in general one will not let the template project over the image borders, so the computation stops a few pixels short of the image border). Only the values $\bar{t}$ and σ_t can be computed in advance, all others change for every new corner position (x, y) . In our approach from Sect. 1.5 only the defined regions of interest are searched for the templates, not the entire image, but even then a lot of computing has to be done. This is especially true if not only a single type of object is to be detected but several classes of objects, each class represented by several templates to compensate for small deviations in the appearance of the objects. It stands to reason to look for ways to optimize this procedure, make it a little more intelligent. Before we come to optimization methods for the matching procedure, we want to discuss some properties of the algorithm one should be clear about before applying it.

Properties: Figure 4.13 shows an image with several simple shapes. For clarity's sake no intermediate gray levels have been used, only white and black. The figure further shows the correlation of this image with the second figure from the left in the top row, the cross, used as the template. White corresponds to a correlation of 1, black to -1 , medium gray to a correlation of 0. It can be seen that:

- After rotating the cross shape by 45° , it does not correlate better with the template than the other shapes.
- Only at a single position, when the template is positioned precisely over the original object, do we have a white pixel in the correlation image, i.e. a correlation of 1. Next to this position the correlation drops rather sharply.
- There is some correlation between the template and the “wrong” shapes, most significant in the case of the outlined square, which can be quite similar to the cross at some positions due to its long, narrow outer edges.
- There is considerable correlation in the surroundings of some shapes, most obvious for the solid square. The reason is that the more extended and filled a shape is, the larger the overlap between shape and template will be. This leads in

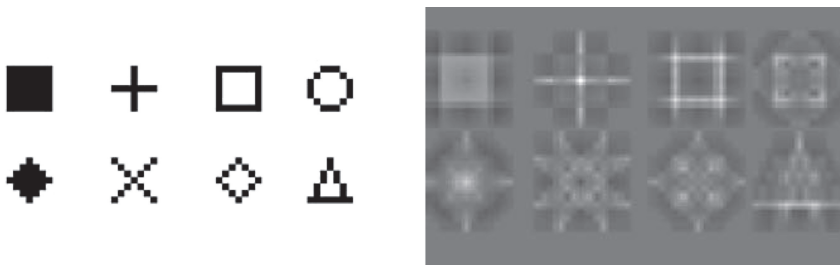


Fig. 4.13 Simple shapes and their correlation with straight cross as template

turn to gradually and continuously increasing correlation in the vicinity of the shape. This effect is of course much more pronounced for higher resolution images and more extended shapes.

What do these examples show? As can be seen from the poor correlation of the straight cross with the diagonal cross, template matching is obviously quite sensitive with respect to distortion or rotation of objects. Of course this is an extreme example, with respect to the type of object as well as to type and extent of the alteration, but we have chosen it on purpose to demonstrate this basic problem of template matching.

Another important aspect is the type of objects. The cross with its narrow arms will give significant correlation only in a very small neighborhood of the object position whereas the filled square, because of the much larger overlap, will show high correlation coefficients at a much greater distance. On the other hand, the cross is more selective as a template, precisely because of the smaller overlap.

Especially important for optimizations of the algorithm is the observation that correlation typically does not change abruptly but rises gradually in the neighborhood of an object. This gives rise to the possibility of searching the image in steps larger than one pixel for candidate positions and scanning their neighborhood more thoroughly.

In summary template matching is well suited for extended, solid objects, less well suited for intricate or fine objects, and is very sensitive to geometrical changes.

4.4.2 Optimizing Template Matching

The basic template matching method described above performs a full correlation with all points of the template for all image pixels. This entails an enormous amount of computation. It will be advantageous to increase performance by a combination of skillfully constructed algorithms requiring less computational effort.

Coarse-to-fine-search: The properties of template matching demonstrated above can be used as starting points for the optimization. We saw that correlation will already show a marked increase in the neighborhood of an object, the stronger, the more solid the object is. Instead of scanning the entire image pixel for pixel, one could therefore move the template in larger steps across the image and perform a pixel-wise search only in areas with high correlation coefficients. This method is commonly called coarse-to-fine-search because of its two stages: a coarse stage looking for candidate positions and a fine stage determining the exact pixel position of the object.

Considerable speed-up can be achieved using coarse-to-fine-search. If the step size for the first stage can be increased from one pixel to four, the first stage will require only 1/16th of the original computation time. Of course, the time for the

fine search stage has to be added, but in most cases it is comparably small, according to the significance of the correlation trends and the number of positions to be found.

It has to be emphasized, however, that the applicability of coarse-to-fine-search and the possible step size depend on the properties of the templates. Figure 4.14 shows the result of a search using the straight cross as template, similar to Fig. 4.13. The search was performed with step sizes of 1, 2, 3 and 4 pixels. For a step size of 4 pixels no significant correlation can be observed at all in the neighborhood of the object because the step size is larger than the width of the object's structures.

The results have to be viewed with the qualification that the straight cross is a rather extreme example for a template. In real images, correlation trends are usually much less abrupt, so image resolutions which are, for example, suitable for reliable character identification allow for a considerable increase in step size. However, the basic principle remains the same: the possible increase of the step size becomes smaller the more intricate the objects are, i.e. the more they are characterized by edges instead of areas.

Subsampling: Above, we have used an increased step size for moving the template across the image as a first optimization method reducing the number of correlations to be performed. However, every single correlation still requires the same amount of computation as before (Eq. (4.2)). This computation effort can be reduced in a similar way and based on the same reasoning. The comparatively continuous changes of the correlation across the image make it possible to use only every second, third or fourth point within the template instead of every single one. The effect on computation time is also similar to that of a coarse-to-fine-search. The same holds for the results and the applicability.

It is quite possible to use both methods simultaneously. When both methods are combined it will typically not be possible to increase the step size as much as when all pixels of the template are used or to sample the template as sparsely as for a step size of one, but this does not mean that the effects have to cancel each other out. Since the possible step size depends mainly on the properties of the image that is scanned for objects, whereas the possible subsampling depends primarily on the properties of the template, the speed-up achievable by the combination of both methods can be higher than when both methods are used in isolation. Figure 4.15 shows the search results for a subsampling factor of two and step sizes of one and two. A subsampling factor of two corresponds to a speed-up factor of approximately four,³ a step size of two results in the same speed-up so that the combination allows for an acceleration by a factor of 16. Using only a coarse-to-fine-search, this speed-up would require a step size of four pixels, which did not allow for object detection at all, as shown in Fig. 4.14.

³ There will of course be some inevitable overhead, but for a comparison of the methods one can roughly assume that speed-up is linear to the reduction of the number of points.

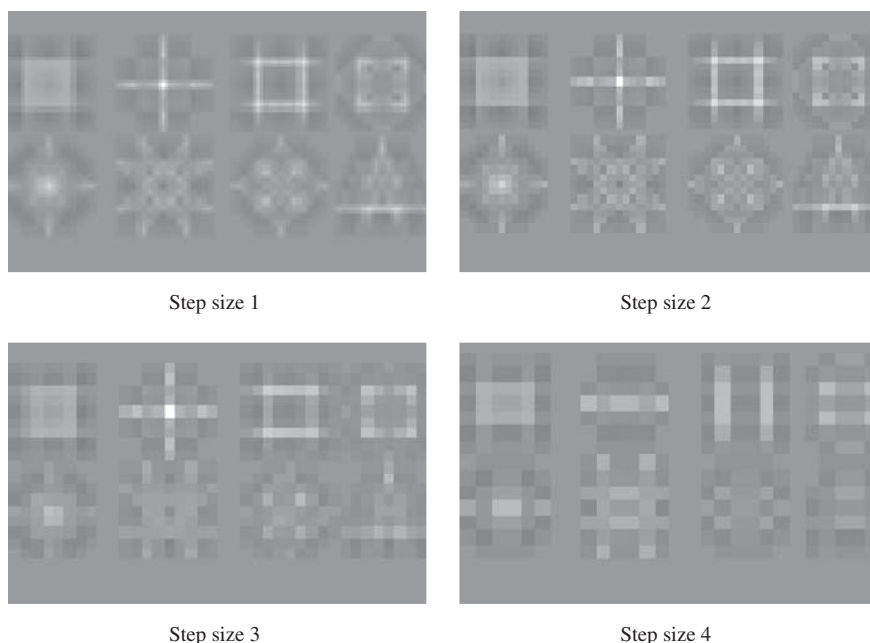


Fig. 4.14 Search for straight cross with various step sizes

Optimization of pixel positions: When using the simple equidistant subsampling of templates introduced above, one may encounter the problem that the points used for computing the correlation may be in very unfavorable locations. For example, a correlation point located on an edge will contribute to the correlation coefficient only if it is positioned precisely. With coarse-to-fine-searching, it may occur that such a correlation point always misses a position where it would contribute to the similarity measure. This problem can be remedied by optimizing the positions of points used for the correlation using appropriate criteria. We cannot cover the various methods which can be used for such an optimization problem, e.g. self-organizing neural networks or combinatorial optimization algorithms. Depending on the application, one can optimize for high correlation with only the respective template or also for good differentiation from other templates.

As an example, Fig. 4.16 shows the correlation points for the detection of a digit printed on rubber. In this application it was possible to accelerate the matching process for these rather fine patterns by a factor of four and simultaneously increase the matching quality, i.e. the average value of the correlation coefficients for the correct object positions, by placing the correlation points at more significant locations in the template.

Truncating the calculation: Finally, it should not go without mention, that there is quite a simple method to accelerate correlation from which all the above optimization methods will benefit. It consists in aborting the computation as soon

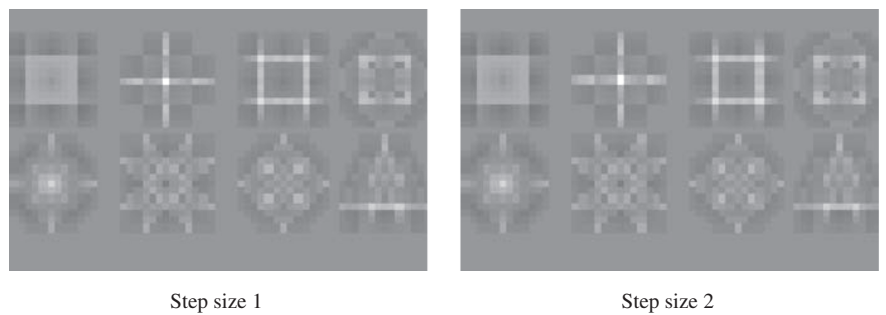


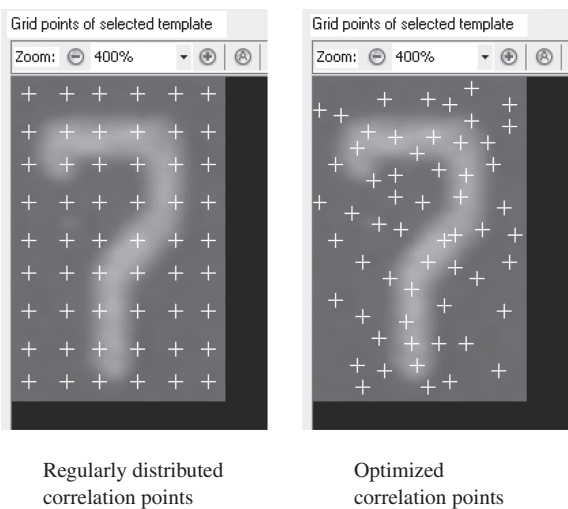
Fig. 4.15 Search for straight cross with subsampling two

as a failure becomes obvious. Typically, there are many more locations in the image where matching will fail than locations where something will be found. If early recognition of such a failure were possible, a considerable amount of computation could be saved at most locations in the image.

A first method for early recognition of a matching failure follows from the fact that the denominator in Eq. (4.2) is always positive. A good candidate position for the presence of an object is indicated by a high positive correlation, i.e. when the numerator of the correlation coefficient becomes negative, there will certainly not be a positive correlation, much less a high one, at this position and the algorithm can be terminated before the computation of the standard deviation σ_g .

Another possibility is based on the customary practice of requiring a certain minimum quality for the matching. Usually, candidate positions with a correlation below a certain threshold (often 0.5) will be rejected. If computation of the numerator of the correlation yields a positive result, the maximum permissible σ_g in the denominator can be immediately computed so that the correlation coefficient

Fig. 4.16 Optimizing correlation points



stays above that threshold. This yields very simply the highest permissible value for the expression

$$\sum_{i=0}^{b-1} \sum_{j=0}^{h-1} g_{x+i,y+j},$$

occurring when computing the standard deviation. As soon as the summation reaches this value, computation can be aborted since the final value of the denominator will always be larger, reducing the correlation coefficient to a value below the threshold.

4.4.3 Comments on Template Matching

The above discussion may give the impression that template matching is a very sensitive and complicated method, raising the question of what it is good for at all. Actually, it is a very powerful segmentation algorithm if configured correctly. This provides the topic of this final section on template matching.

Flexibility: In contrast to contour tracing, template matching can segment objects which are not visually contiguous. This is often the case in character recognition applications. There are letters which are by their very nature non-contiguous, like the “i”. More importantly, there are marking methods which tend to create characters that are always non-contiguous. Examples include the ink-jet technology frequently used in logistics and packaging technology or pin-marking methods often used for car and machine parts because of their durability.

Furthermore, objects may be found using template matching which have no clearly demarcated borders in the image. This can be an advantage or a drawback, depending on the application. For example, area computations for such objects are not very meaningful since only the area of the template itself can be determined.

The “built-in” normalization of the correlation coefficient makes template matching less sensitive to lighting variations than thresholding—provided that the brightness differences between object and background are not reversed. Even then, relatively simple preprocessing can enable matching again.

Program hint: Successful use of template matching requires being able to try all the various optimization options in a fast and flexible way. The check function *Template Matching* in **NeuroCheck** enables you to test all optimization parameters mentioned above (coarse-to-fine-search step size, subsampling, point position optimization) with different settings.

Correlation in the frequency domain: One optimization method has not yet been mentioned, since it is not very important for industrial applications of template matching: correlation after a prior Fourier transform. The complex correlation formula is reduced in the frequency domain to a simple multiplication, but this optimization method is only effective for templates which are so large that the computation effort for normal correlation exceeds that for the Fourier transform.

Generalized template matching: The above mentioned sensitivity of template matching with respect to rotation or distortion of the objects can be reduced by using several slightly different templates for each object. These templates can either be created one by one during the configuration of an application or created automatically during matching from a reference template. This will enable reliable detection of objects with varying appearance, but the computation effort will increase linearly with the number of templates.

Finally, template matching can be preceded by a filtering of the image allowing, for example, for larger step sizes with equal recognition certainties.

4.4.4 Edge-Based Object Search

As mentioned before there are other methods besides template matching that can be used to search known objects within an image. While template matching looks for these objects in a gray-level image, the methods presented here use edge images. Before the objects are searched for, the image is transformed using filter operators discussed in Chap. 2. Figure 4.17 shows the result of an edge extraction using a rather simple operator.

It should be clear that segmentation of objects from such an edge image is far from trivial. Searching for objects in edge images has the advantage that the creation of edges is relatively insensitive with regard to changes in brightness; edge images contain direction information in the shape of gradient direction.

Shape-based matching: Shape-based matching follows the same principle as template matching: The template is moved over the image and at each position template and image are compared. Unlike template matching, correlation is not used as a measurement for the quality of the agreement. Steger et al. (2008) use the sum of the scalar products between the gradient vectors of the image and the template as quality criterion. Thus for each pixel of the template, the scalar product

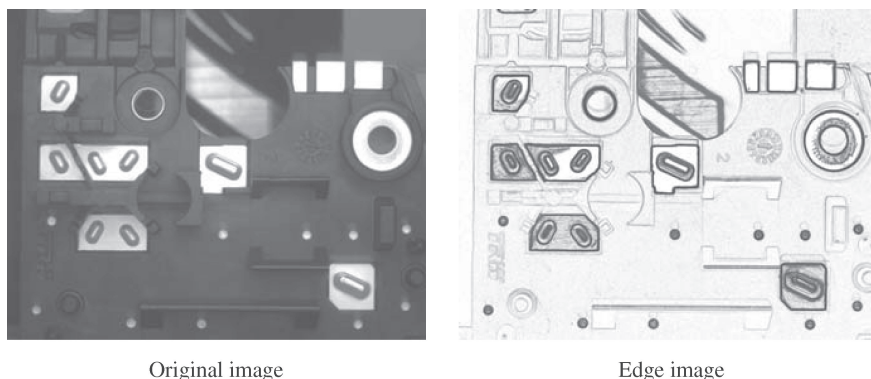


Fig. 4.17 Edge extraction from a gray level image

of the gradient vector with the gradient vector of the corresponding image pixel is computed. The sum of the scalar products over all template pixels yields the quality criterion.

The quality criterion r for the position (x, y) is computed according to Eq. (4.3); please note that g and t are vectors.

$$r(x, y) = \frac{1}{b \cdot h} \sum_{i=0}^{b-1} \sum_{j=0}^{h-1} \langle \vec{g}_{x+i, y+j}, \vec{t}_{ij} \rangle. \quad (4.3)$$

The scalar product is large if the gradients of the template and image run in the same direction and each has a large absolute value, i.e. if there is a discernible edge. If this is the case for many template pixels, the quality criterion for this position is large. In a position where the template does not fit the picture, many edge pixels of the template will be in positions of the image where the gradient is small. This results in the scalar product of the gradients being small. In places where edges in the image and template intersect, the absolute values of the gradients are large but since the gradients run in different directions, the scalar product remains small.

All in all, this method is relatively insensitive to partial coverings of the object and artifacts in the image. To use shape-based matching, many of the optimizing methods used with template matching have to be used here as well. For a detailed description of the method including various optimizations see (Steger et al. 2008).

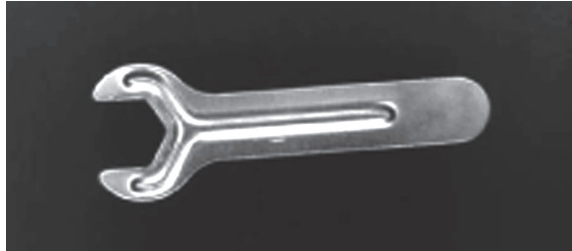
Generalized Hough transform: The generalized Hough transform works exactly reversed from template matching. While template matching checks for each position in the image the quality of agreement between template and image, the Hough transform checks for each edge point what object position it fits to. When a large amount of edge points agree with an object position, then an object has been found.

For the object search only edge points are of interest; in the most simple case this means all points whose absolute gradient value is above a certain threshold. Before a search is started, a reference position within the template is determined and a reference table (r-table) calculated. The reference table contains for each edge point of the template the distance between edge point and reference point, and also the direction. The direction to the reference point is given relative to the gradient direction of the edge point.

For the actual object search it is supposed that each edge point is a point of the object for which there is an entry in the reference table. With the data from the table, the position of the point and the direction of the gradient, the position of the object can be determined. For each image pixel all entries in the reference table are perused and it is counted how often each object position occurs. When an object position occurs frequently, there are many edges within the image that agree with the searched object in exactly this position. Thus there is an object in the image at this position.

The basic principle of the generalized Hough transform was developed by Ballard (1981); however, a number of optimizations are necessary to use the

Fig. 4.18 Object for geometric matching



method in industrial image processing. For a detailed description of the method see (Sonka et al. 2008), and (Steger et al. 2008) for a number of process optimizations.

Geometric matching: The last method we are going to introduce is geometric matching. With this method simple geometric shapes, mostly lines, are searched for in the image. It is then checked whether the found lines can be put together to form the desired object. The screw wrench depicted in Fig. 4.18, for example, has six lines: two form the grip, the other four the jaws of the screw wrench.

At first during the object search all lines in the image are searched for. In the next step, each combination made up of six of these lines is checked whether it agrees with the arrangement of the six lines of the template. If the arrangement is identical, the wrench has been detected; otherwise, the next arrangement will have to be tried. Because of the numerous combinations of the found lines, the computation effort of this naive approach is extreme.

With the appropriate optimizations, this method is nevertheless well-suited to find objects with long edges quickly. However, to find delicate objects with many small edges, the previously described methods, shape-based matching and generalized Hough transform, are better suited. For more details on geometric matching see (Steger et al. 2008).

4.5 Edge Detection

The previous sections of this chapters dealt with locating objects. This section is going to describe how to determine precisely the position of edges whose approximate position is already known. The accuracy of the position determination is necessary since these edges are often the basis for subsequent gauging.

4.5.1 Edge Probing in Industrial Image Scenes

With this approach, the approximate position of the edge is already known. The objective is then “simply” to locate the edge as precisely as possible. Instead of converting the entire image into an edge image as in Sect. 2.3.4, one typically uses

an edge model to directly determine the location of only the required edges. Not only does the computation require far less effort than when using edge filters; it is also much simpler to describe the properties of the edges to be located using an edge model than by designing an appropriate edge filter.

Searching for edges using an edge model is typically performed along search lines in a defined direction. The criteria for the detection of an edge are derived from the gray level profile along the search line. A common edge model has the following parameters:

Edge direction: The distinction between the two possible edge directions, rising and falling, is based on the shape of the gray level profile. If the gray levels along the search line change from light to dark, we speak of a falling edge, otherwise of a rising edge.

Edge height: Acceptance of a gray level transition as a valid edge requires a certain minimum difference between the gray levels along the search line. This is called the (minimum) edge height.

Edge length: The (maximum) edge length gives the distance over which the minimum difference between gray levels indicated in the edge height must occur.

Figure 4.19 illustrates the parameters using a valid and an invalid falling edge. The gray level profile of the invalid edge shows an even larger difference of gray levels than the valid edge, but the distance between two points having the required gray level difference is too long. The edge is not sufficiently steep. The combination of these two parameters, edge length and height, allows for great flexibility in the distinction between significant object edges and gradual changes in gray level resulting, for example, from illumination gradients.

Program hint: The **NeuroCheck** check function *Create Edges* uses an edge model to locate edge positions with subpixel accuracy.

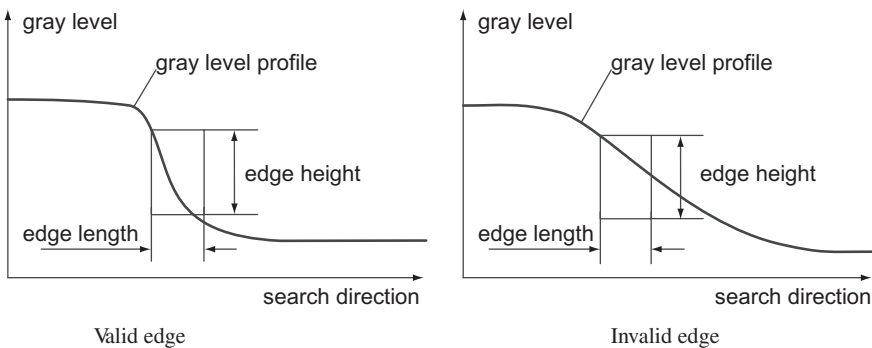


Fig. 4.19 Detection of edges using an edge model

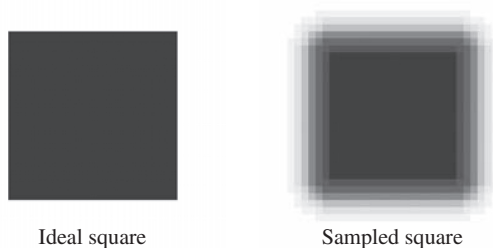
4.5.2 Edge Search with Subpixel Accuracy

In Sect. 1.4.2 we mentioned that the information lost through sampling and quantization can to some extent be recovered. The individual sensor elements of a CCD have a finite size which means they collect light from a certain area of the image scene. In the resulting gray level image, every visible edge will be located exactly along the borders between neighboring pixels. However, the “true” location of an edge in the original scene will almost never coincide with the border between two sensor elements. Instead, it will likely pass somewhere through the pixels. Every sensor element thus receives light from both sides of the edge, sometimes more from the bright area, sometimes more from the dark area. Figure 4.20 demonstrates how this translates into a gradual change between the light and dark area in the gray level image of a black square. Figure 4.21 shows a cross section through the square with gradual edge transition. After interpolating the edge profile, the likeliest approximation of the true edge position is given by the inflection point of the interpolation function, which is also depicted in Fig. 4.21.

It is, of course, possible to interpolate an entire image in this way, but it is usually impractical due to the time and memory requirements, and also unnecessary. For gauging purposes it is sufficient to locate only those edges which are relevant to the gauging task with subpixel accuracy. After the position of the object has been determined through prior knowledge or segmentation, we can probe for the edges and then perform the above interpolation to locate the edge with subpixel accuracy. Distances computed between such edges will then also be subpixel-accurate.

Subpixel algorithms allow for an accuracy of $1/20$ th of a pixel under laboratory conditions. Such precision requires extreme diligence with respect to lighting and image capturing. We will discuss this in more detail in the sections on gauging and illumination, Chaps. 7 and 8. Under production conditions it is usually impossible to guarantee sufficient environmental stability so that an accuracy better than $1/5$ th—or at best $1/10$ th—of a pixel can hardly be achieved. Absolute accuracy depends, of course, on the field of view and the image resolution and—in extreme cases—on the light wavelength, for with appropriate cameras and optics, resolutions in the range of tenths of microns can be reached, where light wavelength and diffraction effects play an important role.

Fig. 4.20 Gradual edge transition due to spatial and brightness discretization



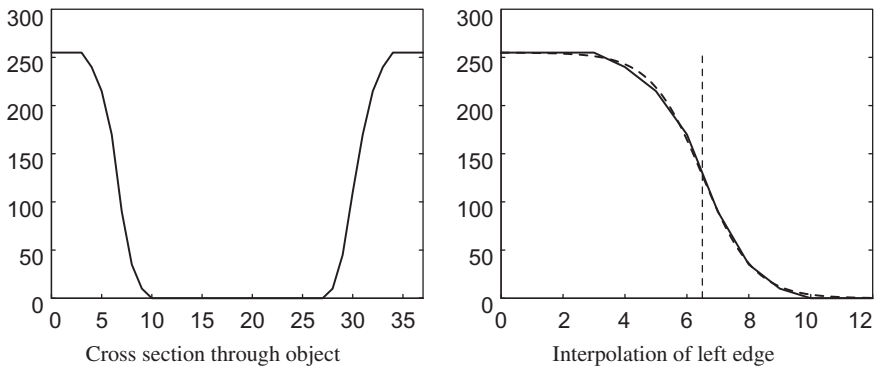


Fig. 4.21 Principle of locating an edge with subpixel accuracy

Program hint: **NeuroCheck** allows for subpixel-accurate detection of edge positions in two different ways:

- Function *Create ROIs by Thresholding* offers an “Edge only” option, which we have used, for example, to create the thread contours in Sect. 1.6. When this option is selected, **NeuroCheck** performs a subpixel interpolation for the point where the search line intersects with the edge. All measurements referring to that point can be computed with subpixel accuracy.
- Function *Create Edges* has already been mentioned in connection with edge models above. It does not use a thresholding process as starting point. Instead it detects significant gray level changes along a search line directly. The location of the edge is then determined with subpixel accuracy by interpolation .

4.6 Summary

Segmentation: Denotes the isolation of objects from the image scene for further analysis.

Region of interest: Regions of interest, abbreviated ROI, also called *areas of interest (AOI)*, limit the image section to be analyzed. They can be used to introduce prior knowledge about the approximate position of objects into the system and reduce the amount of computation since the entire image does not have to be processed.

Object: Denotes a segmented image area corresponding to an actual object in the captured scene. In the framework of our solution approach, objects and regions of interest are largely interchangeable. On the one hand, this enables direct evaluation of fixed areas without segmentation; on the other hand, it directly allows for hierarchical searching for objects inside of objects.

Hierarchical search: In technical applications, objects have to be frequently searched for within other objects; for example, screws have to be detected within the area of a casing. In such cases it is advantageous to be able to search for objects within another object directly. In our example the segmented object “casing” was used as the region of interest for the segmentation of the screws.

Histogram: Table of the frequencies of the individual gray levels inside an image or image area, typically displayed as a bar chart. A histogram analysis can be used to determine an optimal threshold for segmentation by thresholding.

Threshold: Segmentation by thresholding differentiates between objects and background using the absolute brightness value. The gray level ranges of objects and background are separated by the threshold.

Threshold determination: The threshold separating objects and background for segmentation by thresholding can be set to a fixed value or computed from an analysis of the brightness characteristics of image areas. A common method is the bimodality test, which places the threshold between two significant peaks of the gray level histogram.

Local thresholds: In many cases images exhibit different brightness characteristics in different areas, requiring locally different thresholds for segmentation.

Range thresholding: With this method, the gray levels indicating object areas can not only be below or above a single threshold but also within a certain range. Range thresholding allows, for example, for the simultaneous segmentation of dark and light objects on a background of medium brightness.

Connectedness: There are two different definitions of connectedness. Four-connectedness treats only vertically and horizontally adjacent pixels as connected, eight-connectedness also includes the diagonally adjacent pixels. Both definitions have specific advantages and problems for the representation of objects.

Contour tracing: The generation of objects by means of contour tracing is based on thresholding. Starting from a pixel at the border of an object the algorithm moves to the next border pixel according to the definition of connectedness. This continues until the algorithm reaches the starting point again. The method will thus always create closed, contiguous objects.

Contour representation: Object borders created by contour tracing are often represented in a chain code, which represents all possible directions of the connection from one contour point to the next by a single numerical value.

Template matching: This method segments objects by direct comparison of image sections to stored patterns (templates). A commonly used measure of the similarity between image and template is the correlation coefficient. The method is computation-intensive and sensitive to rotation of the objects, but it enables segmentation of objects that are not visually contiguous, which is not possible with contour tracing. Furthermore, the correlation coefficient is quite robust with respect to illumination variations.

Coarse-to-fine-search: Template matching can be accelerated by scanning the image first with a step size of more than one image pixel. Areas of high correlation are then scanned pixel by pixel.

Template optimization: This is a variation of the subsampling method for accelerating template matching by reducing the number of pixels used for correlation. Point positions optimized by combinational algorithms or self-organizing neural networks typically lead to better correlation results than equidistant subsampling.

Edge-based object search: Using appropriate filtering an image can be converted into an edge image. To find objects in an edge image, various methods can be used such as shape-based matching, generalized Hough transform or geometric matching.

Edge probing: This denotes the detection of a relevant object edge along a search line using an edge model requiring a minimum height and slope of the gray level profile along this search line.

Edge extraction with subpixel accuracy: Visible edges in a digital image can only run along the borders of image pixels. This will usually not correspond to the true location of the edge in the image scene. By interpolating the gray level profile the original edge location can be determined with an accuracy better than one image pixel, provided that lighting and image acquisition conditions are kept very stable.

References

- Ballard DH (1981) Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recogn* 13(2):111–122
- Bronstein IN, Semendjajew KA, Musiol G, Mühlig H (2005) *Taschenbuch der Mathematik*, 6th edn. Harri Deutsch, Frankfurt
- Freeman H (1961) On the encoding of arbitrary geometric configurations. *IEEE Trans EC* 10(2):260–268
- Gonzalez RC, Woods RE (2008) *Digital image processing*, 3rd edn. Pearson Education, Upper Saddle River
- Nischwitz A, Fischer M, Haberäcker P (2007) *Computergrafik und Bildverarbeitung*, 2nd edn. Vieweg, Wiesbaden
- Parker JR (1994) *Practical computer vision using C*. Wiley, New York
- Russ JC (2007) *The image processing handbook*, 5th edn. CRC Press, Boca Raton
- Sahoo PK, Soltani S, Wong AKC, Chen YC (1988) Survey of thresholding techniques. *Comput Vis Graph Image Process* 41(2):233–260
- Sonka M, Hlavac V, Roger B (2008) *Image processing, analysis, and machine vision*, 3rd edn. Cengage Learning, Stamford
- Steger C, Ulrich M, Wiedemann C (2008) *Machine vision algorithms and applications*. Wiley VCH-Verlag, Weinheim

Chapter 5

Mark Identification

Identification of work pieces, products, and materials is indispensable in automated production processes, be it for logistics, process control or quality management. There is a variety of methods available for labeling products, which are by no means limited to visually recognizable markings. Magnetic data carriers are a simple example of a non-visual labeling method often used for process control. Non-visual methods are actually easier to identify by automated systems.

There are several things to say in favor of visual labeling methods, however:

- Usually, non-visual labels cannot be identified by humans whereas visual labels can be designed for use by humans and automated systems.
- Non-visual labels often require additional devices to be transported together with the work piece whereas visual labels can usually be put on the work piece itself.
- Non-visual labels can often only be attached to the work-piece carrier, not to the work piece itself. This creates a subtle risk because it has to be ensured that the work piece carrier actually contains the work piece the label refers to. Otherwise, using the work piece carrier with the attached label for process control can lead to assembly errors going unnoticed.

So there are good reasons to use visual labels. In this chapter, the most important labeling methods will be discussed, i.e. bar codes, DataMatrix code and plain text. It will also be shown that other visual labeling methods may often be treated in the same way as readable characters.

5.1 Bar Code Identification

Bar codes are one of the most well-known methods for labeling products and parts of any kind, not least of all because of their use in retailing. They are not readable by humans, unless the plain text is—as it usually is—printed below the bar code,¹

¹ There are supposedly people who are capable of reading bar codes, but it is probably useless to make this a requirement in a job offer.

but they are rather easily and cheaply identified by machines. However, this is hardly a task for a vision inspection system but rather that of specialized scanners. But there are cases where it makes good sense to perform bar code identification using an image processing system, e.g. if:

- bar code identification is just one of several subtasks
- contrast is poor
- non-standard bar codes are used
- bar codes are printed in a circular form
- curved, rotated, or fragmented bar codes have to be identified.

Bar code identification in an image processing system can, in principle, be performed according to the approach from Sect. 1.5. However, one usually does not try to segment separate objects and put together a readable bar code from these objects. Instead, specialized algorithms are used which directly scan a search area for the characteristic brightness profile of a bar code. These algorithms would contribute only marginally to other image processing applications discussed in this text; therefore, we will only make a few short comments on algorithms and continue with application examples and references.

5.1.1 Principle of Gray-Level-Based Bar Code Identification

The fundamental operation of bar code identification is comparatively simple: a defined region of interest is scanned line by line. The sequence of bright and dark areas of the gray level profile of each line is compared to the characteristics of the bar code type to be identified in order to determine whether a bar code like this is present and what characters it contains.

Of course, this method can be easily generalized. It is no problem to scan the region column-wise to read vertical bar codes or at an arbitrary angle for rotated bar codes. In the latter case, however, it would be advantageous to actually rotate the region using an interpolation method in order to minimize discretization problems. We will later see how to use methods for determining orientation from Sect. 3.3.4 for such an application.

This is basically all there is to say about the principle of bar code identification. For someone who actually has to develop such an algorithm, the problem is of course far more complicated. He will usually not be able to rely on constant brightness steps along the scan lines which means that he must decide whether a change in brightness is still significant or not. Also, he will not be able to rely on the equal width of bars and interspaces, e.g. when bar codes are applied on curved surfaces like bottles. Reading such bar codes requires a dynamic adaptation of the module width. This, in turn, can introduce other uncertainty factors because invalid modules may not be recognized anymore. In such cases it is often more appropriate to use special preprocessing methods to “unwarp” the image—provided one has the

necessary prior knowledge. Other problems are the treatment of image degradations, missing parts, scratches in the bar code etc.

Even in this self-contained and apparently well-controlled area, the development of robust identification algorithms is obviously not easy, especially if reliable recognition has to be achieved in the presence of various types of degradation. We will not go into details of bar code reading algorithms here because they do not help much in understanding other types of image processing applications because of their high degree of specialization. For detailed information on bar codes and DataMatrix codes see, for example, Lenk (2005) or (<http://www.adams1.com>).

5.1.2 Types of Bar Codes

This section is going to present some common bar codes that can be identified by most image processing systems.

EAN codes: EAN codes are used throughout Europe for labeling retail products. They occur both in an 8-digit form and a 13-digit form each with an integrated check sum, but the 13-digit form is much more common. Figure 5.1 shows both types. Note the prolonged bars on both ends and in the middle of the codes. EAN codes have a fixed rule for check sum calculation. EAN codes consist of four interlocking module widths, making their identification rather complicated.

UP codes: The UP codes (Universal Product Codes) in Fig. 5.2 are closely related to EAN-Codes, often with the large end bars missing. Indeed, any EAN scanner is able to read UP codes, which are the U.S. equivalent of EAN codes in Europe. The long, twelve-digit code is called UPC-A, the short eight-digit code UPC-E.

Code interleaved two-of-five: It can only depict numbers and optionally may contain a check sum. In contrast to UPC and EAN, the number of characters is not fixed in this symbology apart from the fact that it always contains an even number of digits. The code is space-saving because it always represents two digits in the interlocking bars and spaces (this is the reason why the number of digits is always even). Unlike EAN and UPC, this code has only two module widths. Identification is therefore very robust, making it a preferred code in industrial environments (Fig. 5.3).

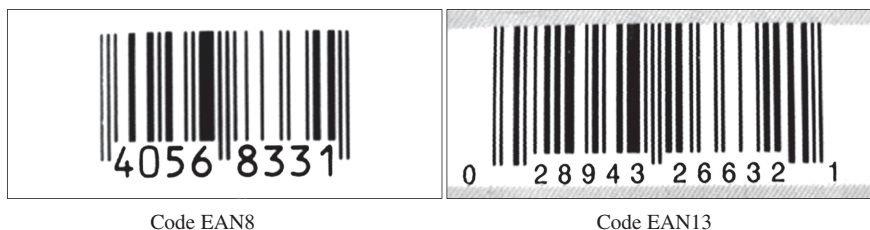


Fig. 5.1 Bar code types EAN8 and EAN13



Fig. 5.2 Bar code types UPC-E and UPC-A

Fig. 5.3 Code interleaved two-of-five



Code 39 and PZN: Code 39 is one of the bar codes capable of encoding alphanumeric characters. Because of its large character set, it requires a comparatively large amount of space and is therefore used only where alphanumeric is actually required. The code can contain any number of characters. Because of its flexible format and character set, it is often used for the identification of parts exchanged between different companies.

The pharmaceutical central number code (PZN code) is the standard code on packaging of German pharmaceuticals. Actually, it is a Code 39 with some additional regulations. It always consists of a minus sign, six digits and a check digit (Fig. 5.4).



Fig. 5.4 Code 39 and PZN

5.1.3 Examples for Industrial Bar Code Identification

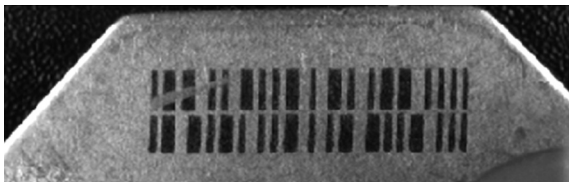
Bar code on metal: Bar codes in an industrial environment are often of far poorer quality than, for example, in retail—and even there scanners fail frequently enough. Degradations are especially common when a bar code is applied directly on a metal surface. Figure 5.5 shows two 2-of-5 interleaved bar codes printed on metal. In this case the bar code damaged by a scratch can still be identified easily because the damage runs across the bars. Such a defect can for example be corrected by averaging over subsequent rows, provided the bar code is oriented exactly horizontally. Even a small rotation can cause blurring of the bar code edges by averaging of several lines. In such a case, one could take undamaged parts of the code from different lines and put them together to form a single unbroken line.

In any case, such a defect will reduce the reliability and speed of recognition because the poorer the quality of the bar code, the more elaborate the algorithms required to make sense of the recognizable fragments. In conjunction with this it may be of interest that the *first read rate* of scanners is typically far below 100 %. What is at most annoying for a cashier can be a real problem in industrial applications since every additional identification trial needs time. Furthermore, scanners usually need more contrast between light and dark areas of a bar code than an image processing system, which can produce faultless identification with contrast levels as low as 5 %.

Program hint: You'll find this example in the download in the directory Examples\Identification. Note the high smoothing parameter in check function *Identify Bar Code* and the fact that every single line of the region is scanned instead of every fifth or even tenth line, which is possible to speed up recognition of a high quality bar code.

Two stage bar code recognition: Sufficient resolution is of prime importance for bar code identification. Usually it will be necessary that a thin bar—as well as a thin space—should at least be three pixels wide. When degradations are to be expected, four or five pixels or more are preferred. Otherwise, the camera will not be able to show the brightness transitions with sufficient distinction. This also means that long bar codes, such as the above mentioned Code 39, require considerable image resolution. With a minimum line width of three pixels, ten characters in Code 39 require 570 pixels which is almost the image size of cameras with low resolution while disregarding the quiet zones to the left and right of the code that are necessary for reliable detection.

Fig. 5.5 Damaged bar code on metal



In the area of warehousing, a two-step approach is frequently used employing a pan-tilt-zoom camera (PTZ camera for short). For example, if a bar code has to be read on a container, it is usually not precisely known at what position the bar code can be found. First, the entire container surface has to be searched for the bar code label. The image field of the PTZ camera has to contain the entire container front. It is, of course, impossible to read the bar code at this resolution, which is why the search first concentrates on finding the entire bar code as one object. From the coordinates of this object the adjustment for the PTZ camera are calculated. After the camera has approached the calculated position, the bar code almost fills the entire image field and can be read without further problems.

Rotated bar codes: We conclude with some remarks on identifying rotated bar codes. Figure 5.6 shows the same bar code in three different orientations. It's a two-digit two-of-five code on a drill for warehousing. The slight rotation in the second image allows for correct reading, the stronger rotation in the third image yields a wrong identification.

In principle, the problem can be solved by determining the preferred direction of the bar code and conducting the scan along this direction. However, only direct pixel information is available using this method. Depending on the resolution, transitions between bars and spaces may be distorted. Instead, the preferred direction of the bar code can be used to rotate an image section with appropriate gray level interpolation. The bar code can then be read correctly.

The preferred direction can be determined using methods from positioning introduced in Sect. 3.3.4. First, the entire drill is detected, then all bright objects of sufficient size on the drill (note that the laser actually marked the background of the bar code, not the bars, which have the same brightness as the drill surface and therefore cannot be segmented as objects). Thus, ten objects are created that can be regarded as a contiguous group, allowing for simple and precise computation of position and orientation. Slight discretization effects can be seen after the image has been rotated, but these do not interfere with the recognition.

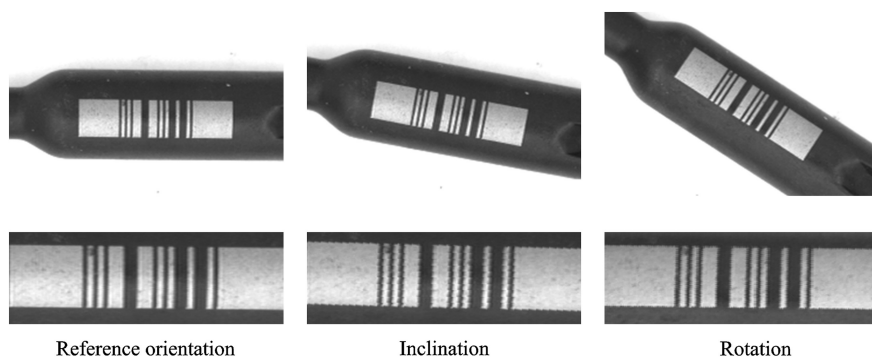


Fig. 5.6 Bar code in different orientations, original and turned back after measuring the orientation

Program hint: You'll find this example in the download in the directory Examples\Identification. The rotation of the image section with the bar code is performed by defining a search line right through the middle of the bar code in reference position. The search line is moved and rotated into the same position and orientation as the current bar code using check function *Position ROIs*. Afterwards, check function *Unroll ROIs* is used to cut out a strip from the image along both sides of the search line and rotate it into horizontal orientation. Note the "Fixed length" parameter in check function *Unroll ROIs*. The diagonal of an image pixel is longer than the horizontal dimension by a factor of $\sqrt{2}$. This would lead to size variations in the resulting image depending on the rotation angle, complicating further processing. The Fixed length option allows for scaling the resulting image to a predefined length.

5.1.4 Two-Dimensional Codes

Strictly speaking, two-dimensional codes are not bar codes because they are not made up of a linear sequence of lines and spaces. However, they are based on the same idea of a geometric arrangement of light and dark spaces, and some types have actually evolved from certain "linear" bar codes. This is why we discuss them here in the context of bar codes.

The importance of two-dimensional codes in industrial manufacturing and especially in logistics is continuously increasing since a much larger amount of information can be encoded using only a small amount of space than by using linear bar codes. Many two-dimensional codes also provide mathematically sophisticated methods for error correction so even damaged code can be read faultlessly.

One of the most common types of matrix codes is the DataMatrix code ECC-200 standardized by the International Standardization Organisation (ISO) in ISO/IEC 16022. ECC-200 is used to differentiate this code from non-standardized, outdated DataMatrix code variants. The code consists of light and dark squares arranged in a square or rectangle with two broken and two unbroken edges as shown in Fig. 5.7. Except for the four edge lines, each square represents one code bit. The eight bits of each code byte are distributed over the entire area of the code so not an entire byte would be lost if the code was damaged.

Depending on the code size, about 40–60 % of the bytes are check information, the rest is useful information. The check information is calculated using the Reed-Solomon algorithm. There are a number of standardized codes to represent a large amount of characters while compressing the data. The highest capacity is available for purely numerical information with two digits being encoded in every payload data byte of the code. Even with textual information it is possible to represent more characters than the code contains payload data bytes by using certain compression methods.

Figure 5.7 shows a 12×12 DataMatrix code laser-engraved in aluminum cast (the edge is usually included in the size quoted). Thus a code area of

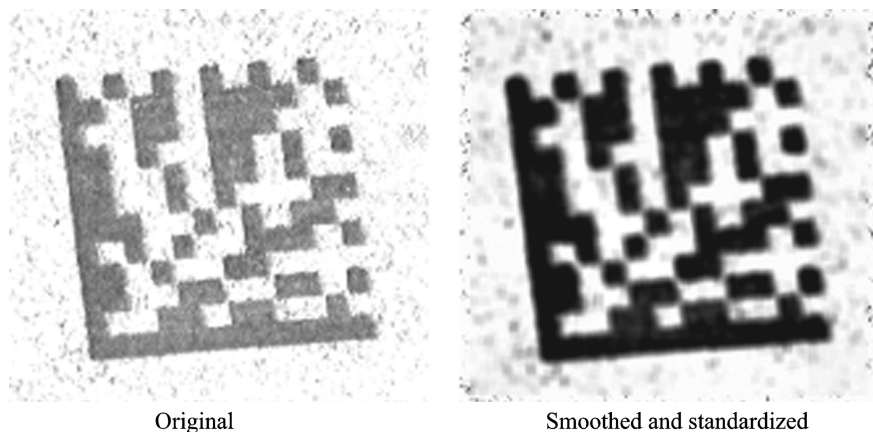


Fig. 5.7 DataMatrix code on aluminum cast

$10 \times 10 = 100$ bit or 12 byte is created. Five of which are payload data bytes and seven correction bytes resulting in a maximum capacity of ten digits.

The image shows a typical background structure for aluminum cast parts leading to considerable fraying of the code edges. The code is best viewed with bright field illumination because the change in surface structure of the code module wrought by the laser reflects the light in a different way into the camera. However, contrast remains poor.

Yet, with this quality the code can be identified. With several preprocessing steps, reliability and robustness can be increased. Smoothing of the image ensures that the surface structure does not cause any interruptions within the code module such as bright spots in dark squares and vice versa. Subsequently the contrast of the image can be improved by standardizing the image with a histogram spread. It's advantageous to proceed with preprocessing in this order since smoothing suppresses white noise that would otherwise be enhanced by standardization. Figure 5.7 shows the code before and after the two processing steps.

After this preprocessing the code is robustly identified even with varying surface quality. Similar to linear bar codes, segmentation and further processing of individual modules as objects is not reasonable because the individual steps of the algorithm are too specialized. Therefore, decoding is usually encapsulated in a function. Table 5.1 shows this inspection task with **NeuroCheck** check functions.

5.2 Character Recognition

Recognizing plain characters that are readable by humans is one of the most demanding tasks in digital image processing. This is true for the segmentation of the individual characters as well as for the next stage in our approach: the

Table 5.1 The following table summarizes the example from Sect. 5.1.4

Category	Description
Index	\\Examples\\Identification
Illumination	Directional front lighting as bright field illumination (camera placed in the direction of the total reflection of incident light)
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i> , a rectangle that encloses the DataMatrix code with room to spare 3. <i>Smooth ROIs</i> 4. <i>Enhance Image</i> 5. <i>Identify DataMatrix Code</i>

evaluation of the segmented objects. In this case, this stage involves the recognition of every object as a particular character.

The available scientific literature fits the difficulty of the task, it is practically overwhelming. However, a large part of this literature deals with the recognition of handwriting, which is of no great importance in industrial manufacturing. Compared to handwritten character recognition, print or other writing produced by a machine has the advantage of stable letter shapes. However, the appearance of the characters can vary even within one batch depending on the application method. Such variations can occur because of wear of printing plates or stamp dies, or simply because of differences in surface structure.

The main difficulty with industrial writing is in the isolation of character from object surface. This step is usually more simple in OCR of handwriting or documents since the writing makes for a good contrast to a homogeneous background like paper. Because of the constant shape, identifying the found characters is usually more simple than handwriting, mostly because of two reasons:

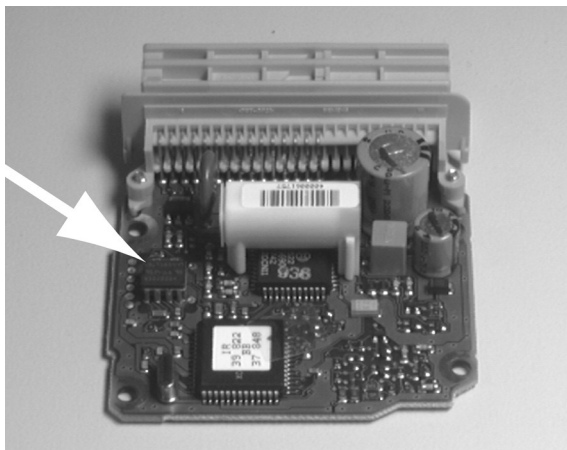
- Characters applied by a machine are written separately and are either not italicized or they have a constant angle of inclination so they can be separated more easily than handwritten characters (however, it is not completely trivial as we will see in Sect. 5.3).
- Machine-applied characters are usually taken from one font so the description method has to capture less variations than with the variety of human handwriting.

This different focus from handwriting will become apparent in the following examples where positioning and segmentation are of highest importance.

5.2.1 Laser-Etched Characters on an IC

Figure 5.8 shows a circuit board for a control module used in cars. One of the inspection tasks was to verify that the correct IC had been mounted at the place

Fig. 5.8 Circuit board from car electronics with IC to be inspected (courtesy of Robert Bosch GmbH, Stuttgart, Germany)

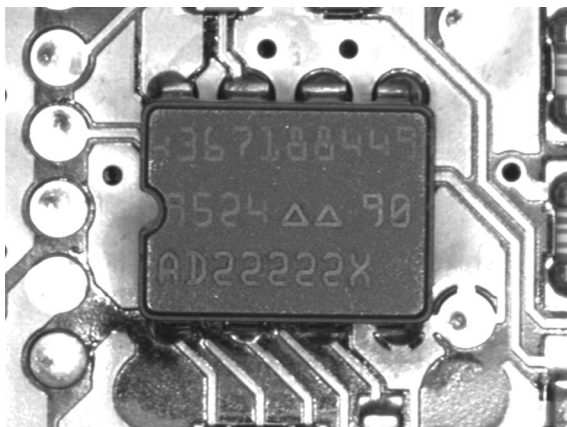


indicated by an arrow. Since this is a safety-relevant part, a 100 % inspection had to be performed.

The characters are laser-etched onto the IC, which is smaller than a fingernail. The IC is black, the writing is dark yellow, almost orange. Contrast is not especially good, which is why strong lighting focused on the IC is necessary.

To make the characters stand out brightly, the light has to be reflected directly back into the camera, suggesting a circular light. This, however, creates the problem of strong reflections, from the circuit board as well as from the IC itself, combined with poor contrast between the characters and the IC. This makes segmentation next to impossible, as can be seen in Fig. 5.9. The solution to the problem is illustrated in Fig. 5.10: cross-polarized lighting. Light from the ring light falls through a polarizing filter. Light rays from specular reflections on the IC will then be completely polarized, vibrating in one direction only. A polarizing filter in front of the camera, oriented at right angles to this direction will block all light from specular reflections, creating a very clear, reflection-free image, shown in Fig. 5.11.

Fig. 5.9 IC illuminated by ring light



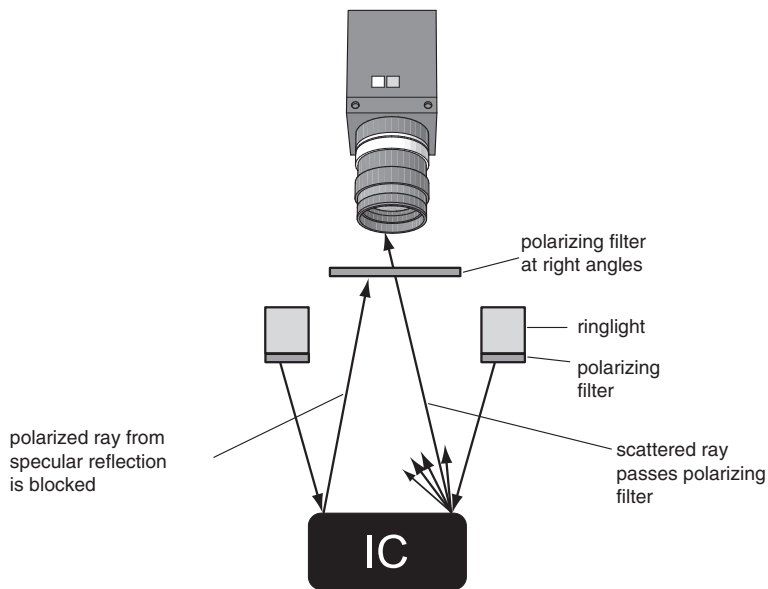


Fig. 5.10 Illumination setup for IC inspection

Fig. 5.11 IC with cross-polarized light



5.2.2 Basic Configuration of Character Recognition

We have already mentioned that character recognition is one of the most complicated tasks in digital image processing; therefore, we will construct this application step by step. We will start with the indispensable elements, then explain the fundamental setup of applications using pattern recognition methods,

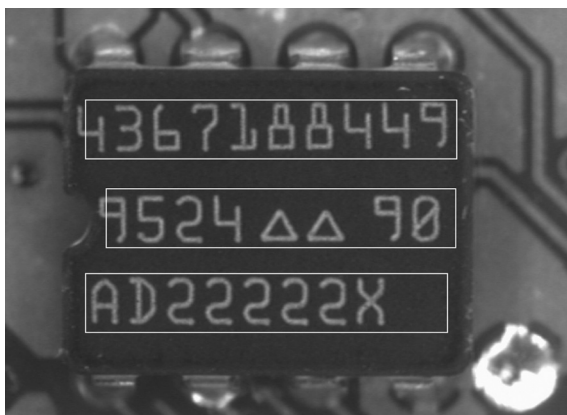
and finally add the refinements necessary for a production application. We will present the setup of the application in the following stages:

1. First we will assume constant and uniform quality of the characters and invariably precise positioning of the IC. Thus we can clearly delineate the basic setup of the application and explain the typical methodology of creating classifier applications.
2. In the second stage we will use methods from Chap. 3 to compensate for the inevitable variations in the position of the IC.
3. In the third stage, we will deal with the problem of varying print quality.

In the following, we will briefly describe the separate steps, then explain the decisive elements of pattern recognition applications in more detail, namely the description and classification of objects.

1. *Image capture.*
2. *Define ROIs:* for each line, a rectangular search area is defined, as indicated in Fig. 5.12. This allows for a structuring of the further inspection in correspondence to the structure of the test piece. It also accelerates segmentation because the system does not have to scan the entire surface of the IC. Please note that the regions of interest have deliberately been defined slightly shorter than the lines. This provides some distance to the remaining reflections from the beveled edges of the IC. The contour tracing algorithm is not affected by this—apart from varying starting points.
3. *Object creation:* since the characters stand out clearly from the background and are not in danger of breaking apart, they can reliably be segmented by contour-tracing on the basis of a binary threshold. Figure 5.13 shows the binary image with the bounding boxes of the segmented characters.
4. *Computation of object features:* at this point, there are two representations available for each character, the image itself and the contour found during segmentation, represented for example by the chain code explained in Sect. 4.3.

Fig. 5.12 Regions of interest for line-wise segmentation



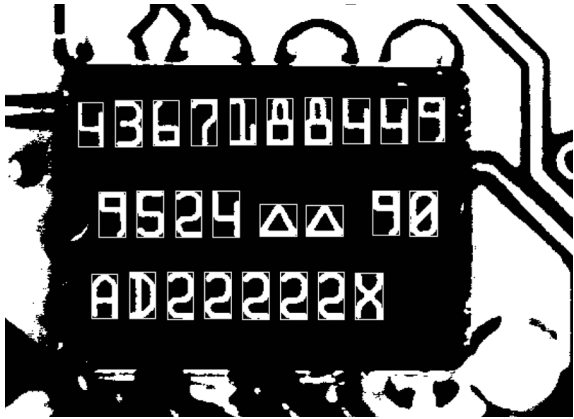


Fig. 5.13 IC with binary segmented characters

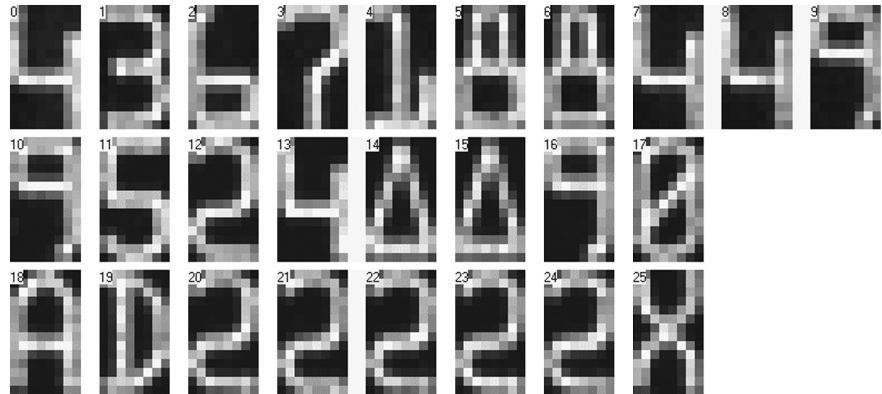


Fig. 5.14 Resampled characters with a resolution of 8×14

Neither representation is well suited for the recognition task to be carried out in the next step. We will see in a moment why this is so and how a useful representation can be created. First, Fig. 5.14 shows the result of the feature computation used in this application: for every character, a graphical representation is created which is normalized with respect to size and contrast, a *resampled image*. The principle is illustrated in Fig. 5.17.

5. *Decision*: in the final step, the actual character recognition is performed by classifying the segmented objects. This means that each object is assigned to one of the possible classes based on features describing it. In this case, the classes are the digits 0–9, the characters A, D, X, and the triangle occurring twice in the central line (Table 5.2).

Table 5.2 The following table summarizes the example from Sect. 5.2.2

Category	Description
Index	\\Examples\\Identification
Illumination	Ring light with cross-polarization to suppress reflections
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i> , by line 3. <i>Determine Threshold</i> 4. <i>Create ROIs by Thresholding</i> 5. <i>Resample ROIs</i> , creates the normalized input images for the classifier 6. <i>Classify ROIs</i> , uses a neural network to assign each object to a class, based on its resampled image

5.2.3 Fundamental Structure of a Classifier Application

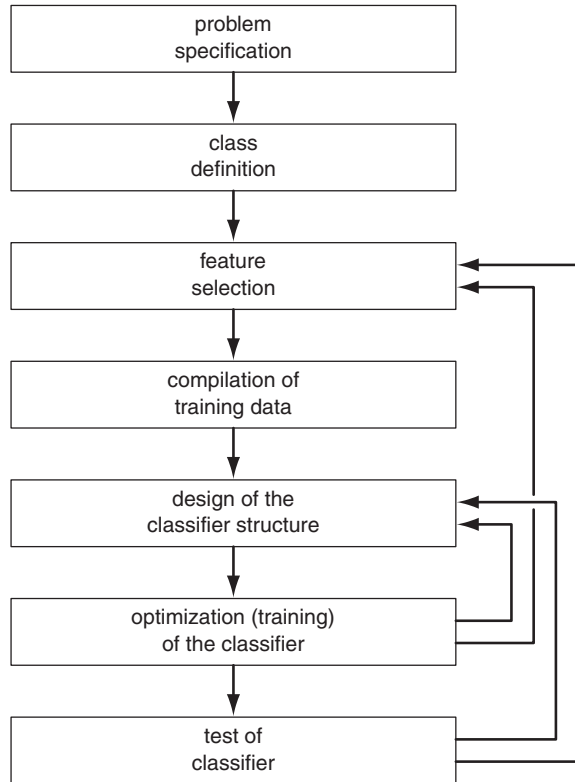
In the preceding section, we simply took for granted that there are methods to create features for classification and to classify objects in order to present the entire application coherently. This section makes up for the omission by giving the explanations necessary for understanding the workings of the check routine. Theoretical foundations for classification systems will be described in Chap. 6.

Procedure for designing classification applications: Figure 5.15 shows the typical design procedure for an application using a classifier such as the character recognition application above.

Now we are going to discuss the individual steps:

1. As always, one has to be clear about the task first.
2. The set of classes to be recognized usually follows automatically from the problem specification. In our example, it is known from the production process that apart from the ten digits only the triangle and the characters A, D and X can occur. This prior knowledge allows for limiting the number of outputs of the classifier to 14, making for a much simpler classifier than if the entire alphabet would have to be recognized.
3. Selecting suitable features is often far from trivial. Extensive research on this topic has been conducted which we cannot present here even in the briefest of summaries. More information on this topic can be found in texts on statistical modeling and classification, e.g. (Smith 1993, Bishop 1995). In practice, one will usually have to select the most appropriate features from the set offered by the image processing system used by experiment, or—if the system allows for such extensions—program additional features suggested by the type of classification task.
4. The next step is the compilation of training data to optimize a classifier. This requires a collection of sample parts, ICs in our example, showing the patterns to be recognized in various qualities. In order to ascertain reliable classification in production operation, the training data set should contain borderline cases of barely acceptable quality. This, by the way, touches on one of the biggest

Fig. 5.15 Procedure for designing classification applications



problems for industrial applications of classifiers. It is often quite difficult to get hold of such a collection of sample parts since one often works with prototypes or parts from the very first stages of production during the design phase of the classifier. This means that the quality of patterns to be classified by the system during production operation cannot be foreseen. It is therefore very important in selecting a system to make certain that training data can be supplemented during operation in order to compensate for process drift and to optimize the system without interrupting production.

5. Next, the basic structure of the classifier can be created. Its input–output structure is already given by the set of classes and features, but most types of classifiers have a number of free parameters affecting their behavior. This step is hard to generalize because type and number of parameters differ considerably between classifiers and systems.
6. Next, the classifier is “trained,” i.e. its basic structure is optimized according to the training data to enable the classifier to solve the given task faultlessly. There are types of classifiers, e.g. the *nearest neighbor classifier*, in which building the structure and training is synonymous, but most sophisticated classifiers use the two-stage process of construction and optimization. If training fails, the first thing to do is to adapt the classifier structure. If this does not lead to a solution,

one can try to change the feature selection, e.g. by adding other features or by removing redundant features which may be hampering the training process.

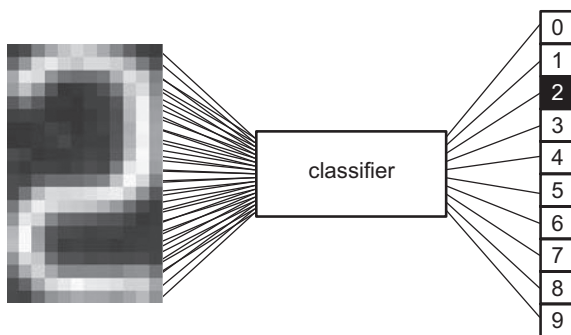
7. After successful training, the classifier should be tested on material not used for training. If its performance in this test is deemed unsatisfactory, one can again try optimizing both points mentioned above, classifier structure and feature selection.

The procedure just described shows that building a classifier application can at times be a quite elaborate process. In many practical cases, however, a little experience will usually lead to satisfactory results quite rapidly. In the following we will make a few remarks on the features and the classifier used in this example.

Describing characters for classification: There are many ways to describe characters—or other patterns—for automatic recognition purposes. Especially in handwritten character recognition, a large variety of features is used to master the range of variations in handwriting, some of them very complex. For the recognition of printed—or otherwise mechanically created—characters, a simple image representation is often sufficient because of the great similarity between characters of the same class. There is one problem, however, due to the way a classifier works.

Chapter 6 will present the theoretical foundations of classifiers; therefore, we will not go into details here. Instead we will simply view the classifier from the outside as a black box which receives certain input signals and produces the class information as output. This can be compared directly to any signal-processing component. In our example, this means that the classifier determines which character is represented by each object from the features of the individual segmented objects. Figure 5.16 gives a schematic representation of this principle using the example of digit recognition. In principle, one could simply use the pixels within the bounding box of the classifier as input signals, as indicated in Fig. 5.16, or the individual elements of the chain code describing the contour of the character. But taking the analogy to an electronic signal-processing component a step further, we can see the problem: such a component has a certain number of inputs which have to be connected to something. However, neither the contour description nor the images of the characters will always contain the same number of values. The “2” on the IC is considerably wider than the “1” and contains a lot

Fig. 5.16 Classifier as a signal-processing component



more pixels. The contour description is even more difficult. Not only will contours have varying lengths, depending on the starting point where the object was first found, contours of identical characters may look completely different.

This means that we have to be sure that the “image” of each character given to the classifier for recognition always contains the same number of pixels. For this reason a size-normalization is carried out, combining adjacent pixels so that ultimately an image with a predefined width and height is created. Figure 5.17 demonstrates this method. Before resampling, the “2” is approximately twice as wide as the “1”, afterwards both are of equal size.

In (Schwenk 1991) extensive experiments with typical industrial labeling methods have shown that a resolution of 8×14 pixels is a good compromise between error rate and processing speed for the recognition of digits. Higher resolution can further reduce the error rate, but under some circumstances it can make the classifier more sensitive because it has to adapt to a higher-dimensional feature space. Like the type of features used, such settings are problem-specific and may have to be adjusted experimentally.

Besides normalizing size, normalizing contrast is usually recommended as well. Sticking with our signal-processing analogy, we can say that signals have limited value ranges. For a gray level image as input signal this is the interval $[0; 255]$. The better this range of values is made use of, the better the processing result—and the easier it will be for the classifier to learn the solution. For example, identical characters with different base levels of brightness are initially quite different signals for the classifier, so it has to learn the class affiliations of these signals separately. After contrast normalizing, the characters will be much more similar, as can be seen in Fig. 5.18. The number of differing signal combinations to be learned by the classifier will thus be considerably reduced.

Fig. 5.17 Principle of resampling characters for classification

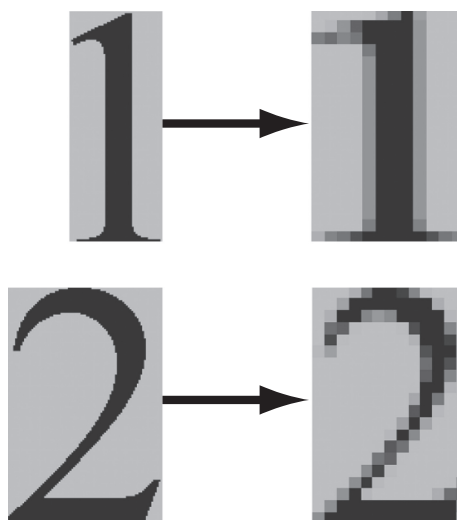
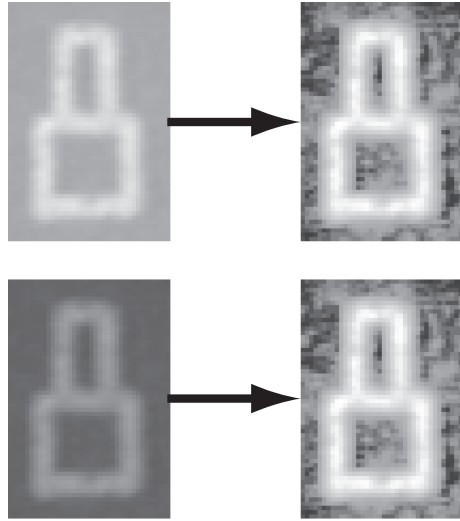


Fig. 5.18 Contrast normalization of characters with different base brightness



Creating a classifier: According to the procedure in Fig. 5.15, creating a classifier requires training data. A training data set contains a number of training samples; each training sample consists of the features of a single object, the resampled image in our example, and the class affiliation of the object. The features can be determined by the system itself, but the class information for the training phase has to be provided by a human “instructor”. After collecting the features of a representative selection of objects, these have to be classified manually—possibly assisted by the system, which can, for example, pre-sort the training patterns by similarity.

The classifier is then created, based on the input–output structure given by the respective number of features and classes, and optimized using the finished training data set. The particulars of compiling training data and creating and optimizing classifiers vary considerably among various image processing systems, therefore we can only refer to the documentation of the respective systems here.

Program hint: The particulars of compiling training data and creating and optimizing classifiers in **NeuroCheck** is described in detail in the online documentation.

5.2.4 Position Adjustment on the IC

Let us return from this excursion into basic aspects of classification to our application example. In Sect. 5.2.2 we assumed—unrealistically—that the IC will always be precisely in the same position. Even if the circuit board is transported on a work piece carrier, this assumption will not hold true, especially if one takes into

account that the IC is about 7 mm wide and stretches across approx. 375 pixels in Fig. 5.11, one pixel corresponds to approx. 0.02 mm. A position variation of a single millimeter will result in a position shift of 50 pixels in the image, making reliable segmentation of the individual characters next to impossible. Rotations were much less of a problem in this application because the transportation method prevented changes of more than two or three degrees in orientation.

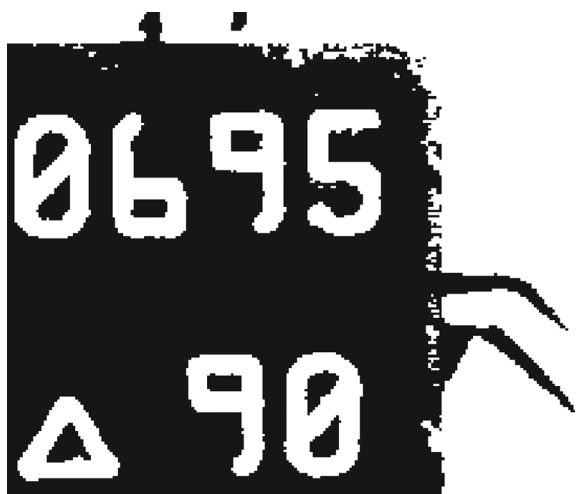
Positioning by edge detection: Because of the dark borders of the path conductors, the body of the IC cannot be easily separated from the background, but the good contrast between the dark material of the IC and the board background in Fig. 5.11 suggests performing the position adjustment in the same way as in Sect. 3.1.2, i.e. using the object edges.

Positioning according to object edges requires a vertical edge for positioning in x-direction and a horizontal edge for positioning in y-direction. The horizontal edge has to be detected right between two of the solder joints at the top border of the IC to form a stable vertical reference. This in turn requires a precise x-positioning first; i.e. a hierarchical position adjustment is carried out in the following way:

1. In the first step, a vertical edge is detected to position the search area for the horizontal edge. Within this search area, the horizontal edge is detected.
2. Both results are used together to position the final search areas for the characters.

The difficulty concerning the detection of the first reference edge lies in the fact that there are practically no clues for its position at first. Depending on the position of the circuit board, a varying number of path conductors can cross the search area so that the check routine has to distinguish between valid and invalid edges. Since the IC as a whole is darker than the path conductors, an appropriate setting for the binary threshold allows for excluding invalid edges from the start, but this runs the risk of fraying the contour of the edge, as can be seen in Fig. 5.19. The reason for

Fig. 5.19 Fraying of the IC edges after removing the path conductors by thresholding



this problem is the beveling of the IC's border which leads to a brightening of the edge in places. Such a frayed edge can hardly be used for a precise position determination. Figure 5.19 also shows the reason why the IC cannot be used for position determination as a whole. Even for such a low setting of the binary threshold the IC still merges with part of the path conductors, especially in the area of the bottom solder points. Such effects are never stable and constant, leading to unforeseeable changes of the IC's shape in the binary image and consequently its apparent position as well.

In Sect. 3.1.2 a smoothing filter was used to stabilize positioning using edges. The same method can be used here to create a sufficiently straight and stable edge, shown in Fig. 5.20.

These refinements allowed for successful position adjustment using the edges of the IC. Over time, however, it turned out that this relatively complex position adjustment procedure, with its narrow parameter tolerances, was too rigid and too trouble-prone. Besides position variations which were too large to detect even the first reference edge, mainly changes in production led to problems. Changes of the board layout made frequent reconfigurations necessary, which required extreme care in this delicate check routine. Changes of the board material resulting from varying market conditions were even more problematic, causing corresponding changes of the contrast between IC and background between different batches. Contrast between IC and board also changed depending on the batch.

Positioning using template matching: Because of the problems with edge detection, a new approach was developed based on the guaranteed presence of the two triangles in the second line. These triangles were used to check the marking at the chip manufacturer's so that the characters are necessarily in a stable position with respect to these triangles. The triangles themselves are eminently suitable for segmentation by template matching, and since—in contrast to the edges—the overall surface of the IC exhibits uniformly constant brightness characteristics, no problems with contrast were encountered here.

Fig. 5.20 Smoothing to improve edge detection



Naturally, the question arises why this approach was not chosen in the first place, in view of the difficulties of the edge-based approach. One should not forget, however, that there is something to be said for using the edges as position reference. On the one hand, template matching is a computation intensive method which may lead to problems, depending on the available hardware and the given cycle time of the process. In comparison, edge detection along a search line is a very fast method. On the other hand, a solution to an engineering problem is always based on a specification to be met. In this case, the original specification did not comprise the beveling of the IC edges nor the possibility of frequent changes to board layout and material. It is not in the least uncommon that a solution which was satisfactory, or indeed excellent, within the framework of the original specification is rendered unusable by production changes and other circumstances, making a new approach necessary. If one has to deal with anything but parts that are produced over many years in absolutely unvarying form, an essential criterion for the selection of an inspection system is how fast and flexible one can react to such changes, how much effort is required for a reconfiguration.

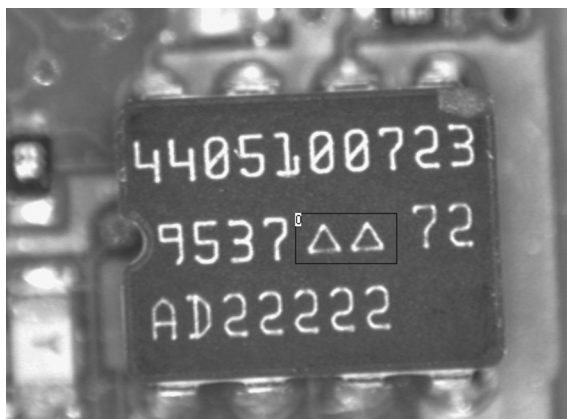
It turned out to be the best approach not to search for two individual triangles, but instead to use the entire pair as a single template. The comparatively large size of this template and the thickness of the lines of the triangles allowed for a large step size for the coarse-to-fine search described in Sect. 4.5.2, keeping the required additional computation time within tolerable limits. The new approach is summarized in Table 5.3.

Of course, the statement from Sect. 4.5.1 that template matching is quite sensitive to rotation of the objects still holds true, but as mentioned above, no changes in orientation of more than a few degrees are to be expected. The segmentation of the pair of triangles by template matching tolerates rotations of up to $\pm 4^\circ$ without problems, as can be seen in Fig. 5.21. The dark frame indicates the position where the template matched the pair of triangles with a correlation better than 0.9.

Table 5.3 The following table summarizes the example from Sect. 5.2.4

Category	Description
Index	\Examples\Identification
Illumination	Ring light with cross-polarization to suppress reflections
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define Regions of Interest</i>, full image 3. <i>Template Matching</i> to detect the pair of triangles as position reference 4. <i>Determine Position of ROIs</i> 5. <i>Define ROIs</i>, by line, at the appropriate positions for the IC in the image used as reference for the positioning 6. <i>Position ROIs</i>, according to the position of the pair of triangles 7. After positioning, the regions of interest will be situated over the lines of the IC in the current image; the rest of the check routine is therefore identical to the procedure in Table 5.2

Fig. 5.21 Template matching on rotated IC



One minor detail has to be noted with regard to a rotation of the IC. Segmentation methods do not necessarily find objects in a meaningful sequence. A human would immediately notice the line structure of the IC labeling; an image processing system, in contrast, does not know anything about these lines. The sequence in which the characters are detected depends on the construction of the segmentation algorithm. For implementational reasons, images—or, more specifically, regions of interest—are usually scanned row by row. In principle, one could of course search parallel to the borders of a rotated rectangle, but this would not only require the rotation to be actually computed, it would also be quite slow because the next point on the search line would not simply be the next image pixel. Instead it would have to be computed from the rotation angle.

Therefore, the characters farther to the right are found first on the rotated IC in Fig. 5.21. Due to the rotation, these characters are higher up in the image so that the horizontal scan lines will intersect their contours first. Figure 5.22 shows the sequence of the characters immediately after segmentation under the given rotation.

Fig. 5.22 Sequence of characters for search by rows



Table 5.4 The following table summarizes the example from Sect. 5.2.5

Category	Description
Index	\\Examples\\Identification
Illumination	Ring light with cross-polarization to suppress reflections
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Enhance Image</i>, provides full contrast within every image section 3. <i>Define Regions of Interest</i>, full image 4. <i>Template Matching</i> to detect the pair of triangles as position reference 5. <i>Determine Position of ROIs</i> 6. <i>Define ROIs</i>, by line, at the appropriate positions for the IC in the image used as reference for the positioning 7. <i>Position ROIs</i>, according to the position of the pair of triangles 8. <i>Determine Threshold</i> 9. <i>Create ROIs by Thresholding</i>, finds the characters scanning the image by rows, i.e. not necessarily in the intuitive sequence 10. <i>Compute Features</i>, determines the x-coordinates of the regions of interest; these are not immediately available after binary segmentation because the starting points of the contours will not usually coincide with the minimal x-coordinate 11. <i>Sort ROIs</i>, sorts the characters within each line according to x-coordinate so that the correct sequence can be evaluated 12. <i>Resample ROIs</i>, creates the normalized input images for the classifier 13. <i>Classify ROIs</i>, uses a neural network to assign each object to a class, based on its resampled image

This character sequence is a problem insofar as a controlling mechanism, e.g. a PLC or an internal function for class evaluation, would not be able to check whether the correct characters are present on the IC in the expected sequence. Of course, it is very simple to solve that problem: the characters are sorted by x-coordinate separately within each region of interest. Incidentally, this is the main reason for the intuitively appealing definition of a separate region of interest for each line. In addition to reducing the area to be scanned, it makes it unnecessary to first distinguish the lines by the y-coordinates of the characters. We will skip the full description of the check routine with sorting at this point because it differs from that in Table 5.4 only by a small detail.

5.2.5 Improving Character Quality

Another problem that became apparent over time is that the labeling process uses two different lasers: the first line and the first half of the second line are written by one laser, the rest of the second line and the third one with another laser. Different age and adjustment of the lasers create strongly differing contrasts, shown in Fig. 5.23. This can lead to problems with binary segmentation, especially in the second line where the contrast changes right in the middle of a region of interest.

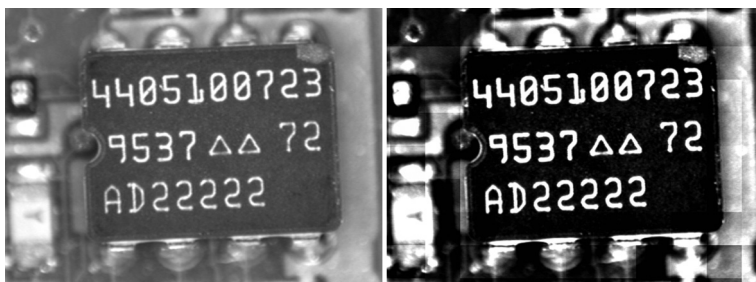


Fig. 5.23 IC with different character contrast, before and after normalization

Even a separate threshold for every region would not solve this problem of a contrast change in the middle of a region of interest.

A possible solution would be to use two regions for the second line. This, however, makes positioning more sensitive. A simpler solution is to use a pre-processing step to normalize local contrast. The image is subdivided into rectangular sections. Within each section, a histogram analysis is performed. Then the gray levels are modified to use the full range of $[0..255]$. Figure 5.23 shows the effect.

Not only does the contrast of the characters differ, line width is also affected. Basically, such a defect could be removed by morphological filtering (see Sect. 2.5). There is one problem however: one would first have to decide where to apply the filter because it should only affect the thin lines. Otherwise the difference would be maintained on a higher level of line width. Unfortunately, one cannot depend on the second laser to write the thinner lines, as Fig. 5.24 demonstrates.

Deciding which characters to filter would require a structural analysis determining the average line width. Such an analysis is not only computation-intensive, it would also require the objects to be found first in order to be analyzed. It is much simpler and more reliable in this case to include characters of both types in the

Fig. 5.24 IC labeling with maladjusted first laser



training set of the classifier. The powerful classifiers available today, for example in the form of neural networks, can cope with this variance without any problems. Table 5.4 summarizes the complete inspection procedure.

5.2.6 Optimization in Operation

It was already mentioned in Sect. 5.2.4 in connection with the problems of edge-based positioning that it is rarely possible in industrial image processing applications to obtain (a) a specification detailing all eventualities and modifications which may occur later and (b) a comprehensive set of sample parts identical to those from serial production. Since today inspection systems are often an integral part of the design of a production line rather than being installed later, serial products will not be available during the development of the image processing solution. It is therefore necessary to develop the inspection using pre-series parts or—at best—parts from production kick-off. Consequently, it is unavoidable to optimize and fine-tune the image processing system during production operation.

To optimize a classifier application in operation requires tracing possible sources of errors, i.e. the image processing system must monitor itself for uncertain results. This is indeed possible, as we will see in Chap. 6. In the above application, all characters recognized by the system with a certainty below an adjustable threshold were automatically collected in a file. From time to time, line personnel looked through this file, removing genuinely faulty patterns that the system was justified to reject. Such faulty patterns were caused, for example, by pollution with dust which can make these small characters unreadable. Such patterns were removed because it is of course dangerous to train a classifier to recognize severely damaged characters, for it may then miss actual errors. The remaining patterns were added to the training data set since they had been rejected only because they differed too much from the patterns used in training, not because they were actually faulty. Then the classifier was trained again with the entire collection of old and new patterns. Thus the system was gradually trained to handle the full range of variations occurring in serial production.

Program hint: Please note that in the example from the download the classifier ICO.CLA was only trained with characters from the image ICO.BMP. This means that it is adapted solely to the appearance of the characters on this one IC. For some characters it has actually been trained on a single pattern. Its performance for characters from the other images is accordingly poor; all characters are recognized correctly, but often with a rather low certainty. You can have problematic characters added automatically to the training data file. Make sure to enable “Automatically save uncertainly classified regions” in the check function *Classify ROIs* and set a rejection threshold of, for example, 60 %. Training the classifier again will improve its performance considerably—on the images from which the newly trained patterns were taken, as well as on other images because the classifier has acquired a broader knowledge of the entire application.

5.3 Recognition of Pin-Marked Digits on Metal

Many production processes require work pieces to be marked in ways resistant to subsequent processing steps. Legal regulations may also demand such durable markings, for example on automotive parts.

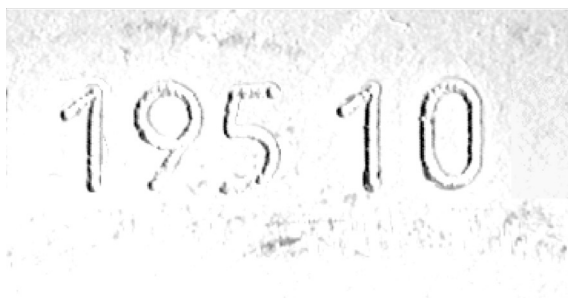
These markings are often applied using marking punches or pin marking machines that create indentations in the material. Because of their three-dimensional structure, characters created in this way are often difficult to illuminate and to segment. Depending on the quality of the marking process, the depth of the characters can vary—even within the same character—causing different reflection properties. This can lead to fragmentation of the characters, which consequently cannot be segmented by contour tracing.

5.3.1 Illumination

Figure 5.25 shows characters applied by a pin marking machine on a metal surface illuminated by simple front-lighting. In some places, no difference or border between digits and background is recognizable, making reliable segmentation virtually impossible.

The problem can be solved by dark-field illumination, illustrated schematically in Fig. 5.26. A smooth reflecting surface will not reflect light into the camera and will therefore appear dark under such an illumination. Indentations in the surface will in contrast appear bright, because edges facing the light source will reflect light into the camera. A heavily textured surface such as created by certain machining methods or caused by pollution will, however, appear bright since it contains a multitude of microscopic indentations which scatter the incident rays so that some of them reach the camera. On such a surface the opposite edges of the indentations will throw dark shadows. Independent of the surface structure, this kind of illumination will therefore cause a significantly different appearance of surface and characters.

Fig. 5.25 Pin-marked digits under simple front-lighting



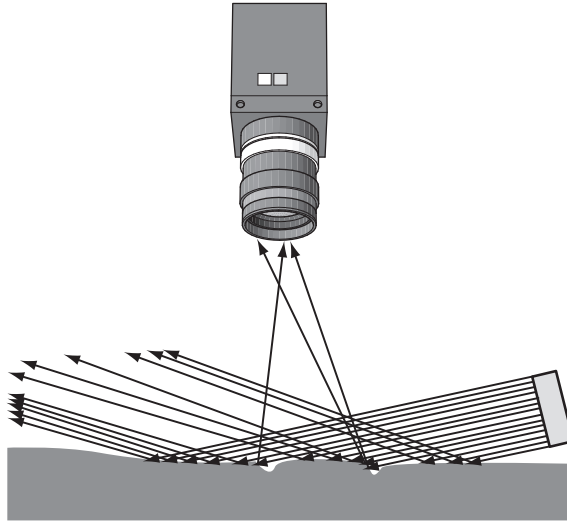


Fig. 5.26 Dark field illumination for pin-marked characters

5.3.2 Preprocessing

The top row of images in Fig. 5.27 shows stamped characters of different quality on different surfaces under a dark field illumination. Obviously, segmentation will still pose a problem despite the fact that the individual indentations making up the digits contrast sharply with the background in each image. The problem is that sometimes the holes in the material appear bright on a dark surface, sometimes the other way around, depending on the depth of the indentations and the texturing of the surface. Since it is impossible to know beforehand which characters—or worse, which parts of the same character—will appear darker than the background and which brighter, it is not possible to segment the characters directly. A pre-processing stage using a contrast filter can solve this problem. The contrast filter emphasizes any differences in the neighborhood of a pixel, independent of the sign of the difference. The bottom row of images in Fig. 5.27 displays the results of filtering with a contrast filter of size 5×5 . Despite the very different source material, all digits appear bright on a dark background.

Program hint: In **NeuroCheck**, you will find the contrast filter as one of the predefined filters in check function *Filter Image*. It sets the brightness of each pixel in the output image according to the contrast in the neighborhood of this pixel in the input image.

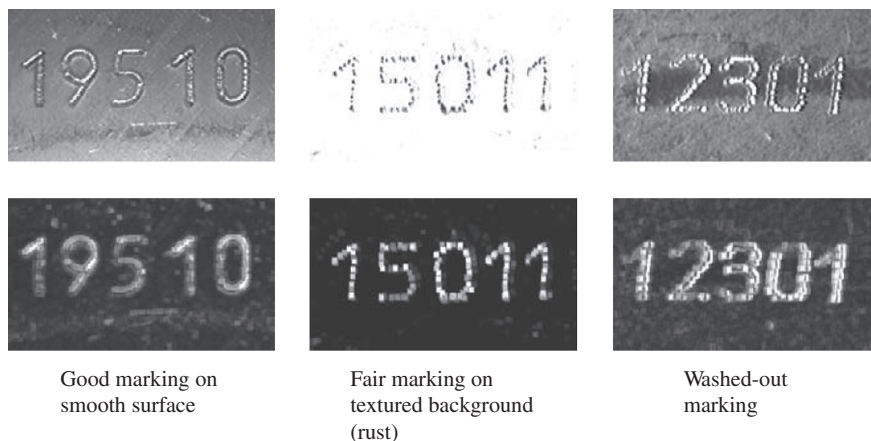


Fig. 5.27 Stamped digits before and after contrast filtering

5.3.3 Segmentation and Classification

After contrast filtering, the digits in all images in Fig. 5.27 appear bright on a dark background, since the contrast on the original surface is minimal. Still the different quality of the stamping causes the digits to appear quite differently even after applying the contrast filter. On the heavily textured surface, which appears bright under the dark field illumination, the individual indentations are small dark dots, whereas the deep, washed-out marking creates thick characters almost merging with each other. In the first case, segmentation by thresholding and contour tracing will not be able to find contiguous contours for each character; in the second it will have great difficulties separating the individual characters. Because of these conflicting difficulties, binary segmentation will not lead to satisfactory results, despite the good contrast. Therefore, template matching has to be used for segmentation. Since the different properties of the characters make separate templates for each of the character qualities necessary to enable reliable detection, it's advantageous to speed up the search. This can be formulated as follows:

Approximate location by thresholding: If the characters have to be searched for within large image sections, the characteristics of the images after difference filtering enable binary thresholding to determine at least the approximate position of the entire group of digits. Relative to this position, a comparatively small search area can be defined that can be scanned much faster by template matching than the entire image.

Dilation filtering: Acceleration of the template matching process itself meets its limits in the small width of the structures of digits stamped with poor quality. Dilation filtering allows for a wider step size of the first stage of the coarse-to-fine search method described in Sect. 4.5.2. Figure 5.28 shows the images from Fig. 5.27 after gray level dilation.



Fig. 5.28 Pin-marked digits after dilation

Table 5.5 The following table summarizes the example from Sect. 5.3

Category	Description
Index	\\Examples\\Identification
Illumination	Oblique dark field illumination to create strong contrasts on the edges of the indentations
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Filter image</i>, emphasizes differences between stamped holes and background brightly, whereas the uniform parts of the surface appear dark 3. <i>Filter image</i>, increases the width of the structures of the digits by gray level dilation to allow for a wider step size for the first stage of the template matching algorithm. Prior to this filtering stage, an approximate location of the character area can be performed by thresholding to restrict the area to be filtered and scanned 4. <i>Define ROIs</i>, encloses the character field within a search area 5. <i>Template Matching</i>, scans the area for digits 6. <i>Sort ROIs</i>, sorts the detected digits by x-coordinate 7. <i>Resample ROIs</i>, creates the normalized input images for the classifier 8. <i>Classify ROIs</i>, uses a neural network to assign each object to a class, based on its resampled image

A considerable part of the inspection time is allotted to the two filtering stages and the template matching which needs separate templates for the different character qualities

Both methods for accelerating the inspection procedure can be used in combination. If the approximation of location is carried out before dilation, an additional advantage is achieved by dilating only the search area for the template matching instead of the entire image. The rest of the procedure is virtually identical to the character recognition on the IC in the previous sections, so we will skip this part here. Table 5.5 summarizes the example.

Program hint: Please note that no patterns from the low-quality image with the fragmented characters on the bright background were used to train the classifier in the example from the download. These digits will therefore be recognized with reduced certainty. The template matching algorithm, however, uses patterns from all image qualities.

5.4 Block Codes on Rolls of Film

Using classifiers must not be limited to character recognition. In principle, any kind of pattern can be recognized in this way, suitable features provided. An example is the identification of light sensitivity codes on rolls of film. In the age of digital photography rolls of film do seem somewhat archaic but, on the one hand, enthusiasts still use them and, on the other hand, this example perfectly illustrates the recognition of non-character-based labeling.

These codes can also be identified in a non-optical way since they are conductive. Miniature cameras recognize the light sensitivity after the roll of film is put in. Figure 5.29 shows such codes with a film speed of 100, 200 and 400 ASA. Identification within the production process, before packaging, demanded a faster non-contact method.

The main challenge in this application was the required inspection speed of 15 parts per second, therefore the hardware setup is the most interesting part here, whereas the inspection procedure appears comparatively simple—if one disregards the complex algorithms beneath the surface of the functional blocks.

Hardware setup: The rapid motion of the film rolls rules out the use of conventional synchronous video cameras with a fixed time for triggering. Furthermore, exposure time has to be as short as possible to avoid motion blur. This, in turn, requires a high-intensity illumination capable of bringing enough light onto the sensor in this short time period—in other words: a strobe.

Figure 5.30 shows the setup. The rolls of film pass a light barrier, setting off a signal which triggers the following events:

- The camera starts image capture and opens the shutter for a millisecond. Then the captured image is sent to the computer;
- After a short delay time during which the camera gets ready for capturing, the strobe light illuminates the passing part for approx. 10 μ s, so that a crisp frozen image of the film roll is created on the sensor.



Fig. 5.29 Rolls of film with 100, 200, and 400 ASA light sensitivity codes (left to right)

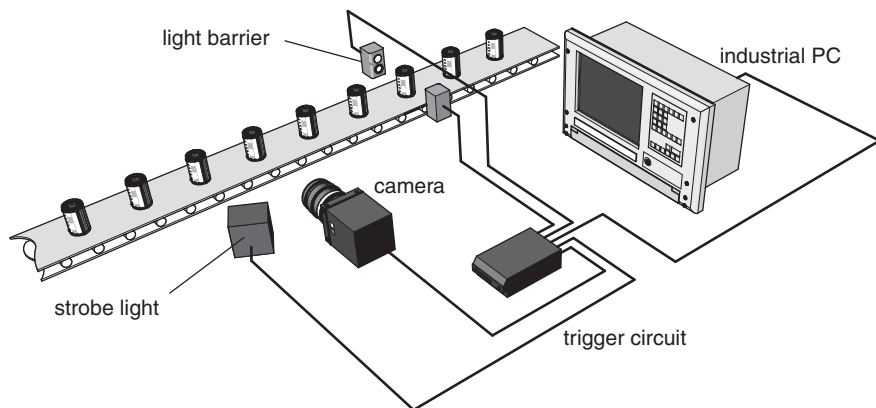


Fig. 5.30 Hardware setup of film roll inspection

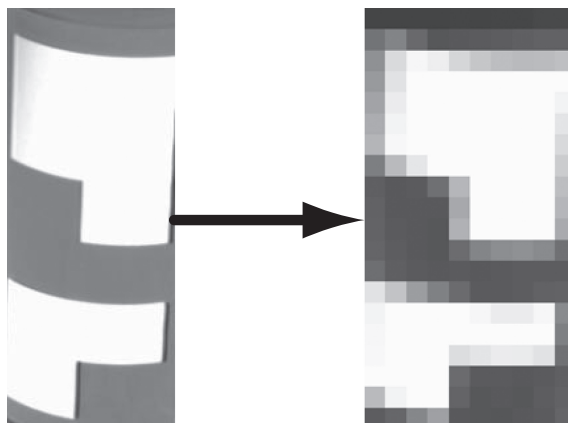
The entire scene is illuminated only as long as the strobe light flashes. This means that the effective exposure time is actually much shorter than the shutter speed, rendering motion blur absolutely negligible. Of course, this procedure will only work correctly with a precise mechanical setup and accurate adjustment of the trigger propagation times.

Inspection procedure: The most interesting aspect of this inspection problem—algorithm-wise—is the application of a classifier to an arbitrary set of symbols instead of characters. An object search is only required here for position adjustment, from which the inspection speed of the system benefits.

The trigger notwithstanding, variations of the position of the film roll within the image in a range of a few pixels will be inevitable because of the high speed. The mechanical setup does not allow for variations in y -direction, so only a single vertical edge is required for positioning. The sharply defined front edge of the film roll was chosen as the position reference. The location of this edge can be determined extremely fast because it only requires searching on a single search line for a transition and cutting out part of the edge as an object. Another advantage is that the strobe lighting creates extremely stable contrast conditions so that no histogram analysis is necessary. The binary threshold for the edge detection can simply be predefined.

Based on the position of the edge, a region of interest is positioned directly over the light sensitivity code. Since this region will—after positioning—contain exactly the code to be read, no object search is required. Instead, the contents of the region of interest can simply be classified. Like the characters and digits in Sect. 5.2.2, the contents of the region of interest are first converted into a resampled image. The main purpose of the resampling in this case is not to ascertain that the number of input values to the classifier is always identical. This would be the case anyway, because the region is manually defined, so it will always have exactly the same size. Classifying a region of this size directly, however, would require a classifier with tens of thousands of inputs since every image pixel represents a separate input signal. Such a classifier would be extremely slow. The main task of resampling here

Fig. 5.31 Light sensitivity code of a 200-ASA film before and after resampling



is therefore data reduction. Figure 5.31 shows the light sensitivity code before and after resampling to 10×20 grid points.

After classification of the light sensitivity code, a second region of interest is positioned according to the position of the determined front edge in order to read the bar code. It is a two-of-five interleaved code with six digits.

Inspection speed: If the combination of camera and frame grabber has partial scan capabilities, capture and transfer time can be reduced by using only part of the image lines. This requires that the camera allows for the partial transfer of image data. If the sum of capture, transfer and processing time is still too large, the camera can already capture the next image while the processor is evaluating the previous image. In this type of parallel processing only the longer of the two time spans—capture plus transfer time or processing time—limits the inspection speed. Table 5.6 summarizes this application.

Program hint: For the parallelization of image capture and processing described above, check function *Capture Image in Parallel* is used in **Neuro-Check** which is responsible for the required synchronization.

5.5 Print Quality Inspection

We conclude this chapter on mark identification with some remarks about an application area dealing with similar objects but with different goals in mind. Print quality inspection is less concerned with reading the marking than with checking the quality of the marking process. This kind of application is common in production areas in which the readability of markings on a large number of identical parts has to be guaranteed, be it for reasons of safety or company image. Typical examples are the verification of company logos, brand names and pull dates in the pharmaceutical and food industries or the inspection of markings on components, switches and instruments in the electronics and automotive industries.

Table 5.6 The following table summarizes the example from Sect. 5.4

Category	Description
Index	\\Examples\\Identification
Illumination	Triggered strobe illumination
Camera	Asynchronous camera, exposure time 1 ms
Triggering	Light barrier, triggers image capture and with minimum delay the strobe
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i> , search line for edge detection for position adjustment 3. <i>Determine Threshold</i> , predefined threshold for acceleration (possible because of stable contrast conditions) 4. <i>Create ROIs by Thresholding</i> , detection of front edge for positioning 5. <i>Determine Position</i> , computes x-coordinate of front edge 6. <i>Define ROIs</i> , rectangle of appropriate size for the light sensitivity code 7. <i>Position ROIs</i> , moves the rectangle according to the edge position 8. <i>Resample ROIs</i> , reduces input data for the classifier 9. <i>Classify ROIs</i> , recognizes the light sensitivity code from the resampled image. 10. <i>Define ROIs</i> , rectangle of appropriate size for the bar code 11. <i>Position ROIs</i> , moves the rectangle according to the edge position 12. <i>Identify Bar Code</i> , reads six characters in code 2-of-5 interleaved

5.5.1 Methods

Print quality inspection methods differ considerably from the identification methods discussed so far in this chapter. Basically, the idea is very simple: an ideal template of the print to be verified is stored; this template does not necessarily have to be taken from a physically existing “master piece”, it could also be computer-generated. Next, an image is created containing all differences between the ideal template and the current test piece. This can be done by subtracting the reference image from the current image.

So much for the idea. Applying this method directly in a real-world context will show very quickly that it will always detect considerable deviations between reference template and test piece, regardless of the actual print quality. This is a consequence of inevitable position variations and inaccuracies of image capture and transfer. A change in position by a single pixel will result in conspicuous print defects along every edge in the image. Figure 5.32 demonstrates this effect. Note that there are absolutely no differences between reference image and test image apart from positioning. There are several strategies for dealing with this problem. In any case, it is a sine qua non to take care that the reference image is positioned with extreme precision relative to the test image (or vice versa).

How to proceed further in a print quality application depends largely on the definition of the often rather woolly term “print quality”, i.e. what defects have to be detected. This is very difficult to generalize because what is a permissible deviation in one application may be a serious quality problem in another. Some

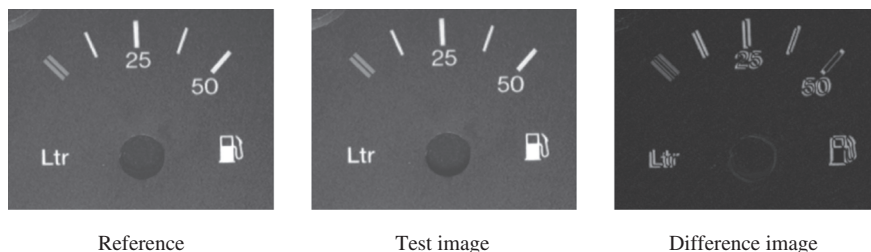


Fig. 5.32 Effect of minor changes in position on image subtraction

possible specifications of acceptable and unacceptable deviations from the ideal pattern are:

Thinning of characters: As long as such defects are within certain limits, they are often tolerable. For example, one or two missing pixel columns on the vertical stroke of a “1” is often acceptable because readability is not affected. In such applications, the inspection system must be capable of ignoring such a deviation from the reference image.

Holes in characters: White spots in a character or logo printed in black will practically always have to be regarded as a defect. The same holds for scratches crossing the characters—which may even break apart in the image due to such a defect.

Spots in the background: This type of problem occurs mainly with ink-jet printers or rubber pad printers with worn out printing dies. This type of error will also be unacceptable in most cases.

It follows from the difficulty of generalizing the types of unacceptable print defects that the inspection procedures themselves will be very much application-dependent because they have to detect and evaluate errors according to the marking method and to the specifications of the client. In many cases, however, the inspection task can be seen as a special case of presence verification in the difference image. This method segments the defects appearing in the difference image as objects and evaluates their properties individually. Such an approach offers considerable flexibility for translating the respective specifications into the inspection procedure. Of course, extreme care has to be taken in the creation of the difference image.

Evaluation of segmented objects in the framework of presence verification is discussed in detail in Chap. 9. Here we will simply give an example for the creation of the difference image and the subsequent evaluation.

5.5.2 Print Quality Inspection in Individual Regions

We will give an impression of how print quality inspection works using the fuel gauge from Fig. 5.32. A possible approach is to define individual regions of interest for every character or logo to be inspected. These regions of interest can be

defined in accordance with the position of the current test piece using position adjustment methods in quite the same way as in the previous applications in this chapter. The circular hole for the axis of the fuel gauge is a suitable reference object for positioning. Despite the poor contrast, this hole can be segmented easily using template matching. In simple cases, this position adjustment may already be sufficient to create the difference image and evaluate it within the search areas.

In general, however, small position variations relative to the reference object will occur within the print itself. This requires an additional position adjustment, performed separately for each region of interest. One can use the same method for this position adjustment as for segmentation using template matching: the reference pattern is moved across the current test image. At every position, the correlation with the corresponding section of the test image is computed until the pattern “clicks into place” at the position yielding the maximum correlation coefficient. In contrast to segmentation with template matching, the search can be restricted here to a small neighborhood of the reference position. Figuratively speaking, the reference pattern is wiggled back and forth until it snaps into position. Figure 5.33 shows the result of this method for a fuel gauge which has not only been moved relative to the reference image as a whole but also contains some actual defects:

- The “r” of “Ltr” is missing.
- The window of the gas pump is partly smeared.
- The 25 liter mark is a little too narrow.

The frames of the regions of interest have been drawn in white in the image of the defective test piece and in the difference image. Obviously, all defects have been detected correctly.

In the next step, the defects can be segmented as objects and their properties evaluated. We will leap ahead here to Chap. 9 on presence verification and Chap. 10 on object features. Suitable features include radii, horizontal and vertical dimension, and area. If the narrowing of a printed item, like that of the 25 liter mark here, is not to be regarded as an error, features like the ratio of fiber length to width can be used, also the average gray level and the contrast, since near the border of a mark the inking is typically thinner, so that errors close to the edge differ noticeably from those in the inner areas of a character.

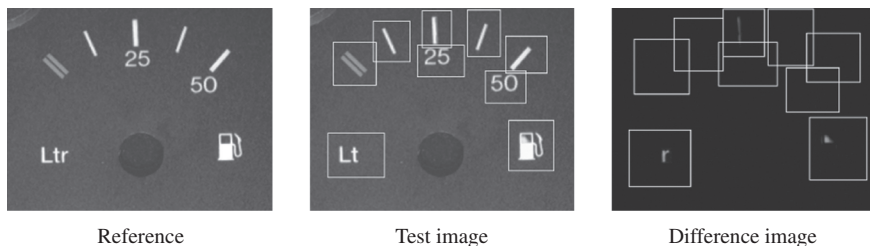


Fig. 5.33 Effect of image defects on image subtraction

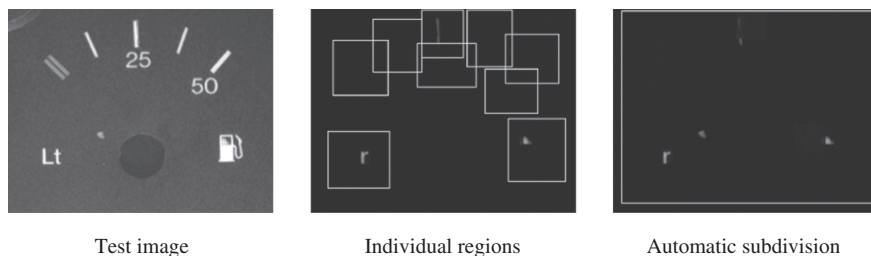


Fig. 5.34 Print quality inspection with automatic subdivision

5.5.3 *Print Quality Inspection with Automatic Subdivision*

A certain type of defect cannot be detected using the approach demonstrated above: additional paint outside one of the regions of interest. If such defects are to be expected, the entire inspection area—in this case practically the whole image—can be defined as the region of interest and subdivided step by step. Each of the new smaller sub-regions can be treated in the same way as the smaller regions of interest in Fig. 5.33; i.e. they are shifted around their original position in small steps until the position of maximum agreement with the image has been found. By setting a division factor (e.g. subdividing into 2×2 or 3×3 sub-regions in each step) and a minimum size of the sub-regions, the method can be fine-tuned to the size of errors to be detected. Figure 5.34 shows the result of both methods when a bright spot to the left of the axis hole is added to the defects described in the previous section. Using separate search areas for the individual characters will miss this defect.

Further inspection can be performed in the same way as before, i.e. as a presence verification. Both methods, global positioning of individual search areas and automatic partitioning of each region, can be combined to provide maximum flexibility for reacting to different types of defects.

This may suffice to give an idea of the typical characteristics of print quality inspection applications. As mentioned before, because of the highly different types of errors, this area can hardly be generalized. Therefore, the evaluation strategy has to be determined for each application. It should be mentioned that print quality inspection methods can be used in seemingly quite different areas, for example inspection of hybrid ICs or printed circuit boards—i.e. generally in any inspection task where the correspondence of the visual appearance of an object with a reference has to be checked.

5.6 Summary

Bar code identification: Image processing systems are more flexible than dedicated bar code scanners and therefore capable of recognizing codes under conditions where a scanner would fail (poor contrast, damaged codes, geometric distortions). Disadvantages, however, include a higher purchase price and considerable development effort for robust reading algorithms, if no ready-made image processing system with built-in bar code recognition functionality is used.

DataMatrix code: Because of the higher density of data and the potential of error correction, two dimensional codes such as the DataMatrix code are frequently used for labeling in industrial manufacturing and logistics. Image processing systems can reliably identify DataMatrix codes even in adverse conditions.

Design of classifier applications: Arbitrary characters and symbols can be recognized by a classifier using appropriate features. The design of a classifier application usually requires the following steps:

1. Problem specification
2. Definition of classes
3. Feature selection
4. Compilation of training data
5. Design of the classifier structure
6. Optimization (training) of the classifier
7. Test of classifier.

Feature selection, classifier design, training and test are typically an iterative procedure.

Sequence of classifier applications: Inspection procedures from this area usually work as follows (disregarding possible preprocessing steps):

1. Definition of search areas, possibly after prior position determination
2. Segmentation using template matching for non-contiguous objects
3. Deletion of invalid objects and sorting, both using simple features
4. Computation of features suitable for the classification, i.e. a normalized visual representation
5. Classification.

Classifier: A classifier can be seen as a signal-processing component with a defined number of inputs for feature values and a number of outputs for the individual classes.

Character representation: A visual representation of characters normalized with respect to size and contrast is usually a suitable feature for the recognition of printed characters. Handwritten characters require more complex features that exceed the scope of this text.

Arbitrary symbols: Classifiers can recognize not only human-readable characters but arbitrary symbols.

Preprocessing: Basic preprocessing methods such as contrast normalization or filtering can ease the task of the classifier considerably by reducing variations in the appearance of the characters or symbols.

Optimization in operation: Changes of problem specification, lack of available samples from serial production during the development phase, changes in production and process drift (e.g. because of wear or misalignment of marking equipment) may require later optimization of inspection procedure and classifier. This requires permanent supervision of the classification certainty.

Positioning using template matching: Under varying contrast and background conditions, detection of characteristic marks by template matching is often more appropriate to determine a position reference than object edges because this method can use a larger number of image pixels.

Non-contiguous characters: Contour tracing requires contiguous characters. Robust segmentation of characters composed of individual dots (for example ink-jet print or pin marking in metal) requires either template matching or filters, which cause the individual dots to merge into contiguous characters but also run the risk that separate characters will merge.

Print quality inspection: Denotes the verification of previously known printed characters or other objects with known appearance such as circuit elements, circuit boards, packaging or product labels. The structure of the inspection procedure is usually that of a presence verification on a difference image. Creation of the difference image requires extremely precise positioning of the reference image or parts thereof. The exact approach depends largely on the respective application and its specifications for acceptable and unacceptable defects.

References

- Bishop CM (1995) Neural networks and pattern recognition. Oxford University Press, Oxford
Lenk B (2005) Optische Identifikation. Monika Lenk Fachbuchverlag, Kirchheim unter Teck
Schwenk M (1991) Mustererkennung mit Neuronalen Netzen. Master's thesis, Institute of Technical Optics, University of Stuttgart
Smith M (1993) Neural networks for statistical modeling. Van Nostrand Reinhold, New York

Chapter 6

Overview: Classification

This chapter will give a brief general introduction into the field of classification and an overview of some important types of classifiers. Classification is a research area in its own right, which over the last 40 years has combined results from disciplines as different as biology, psychology, mathematics, and computer science. Covering such an extensive and varied scientific field with even a minimal claim to completeness is naturally beyond the scope of this text, as is treating all commonly used types of classifiers in mathematical detail. Yet we will at least mention the principal types to give the reader some orientation in this important area of pattern recognition tasks. We will therefore discuss the multilayer perceptron neural network used in the pattern recognition examples of Chap. 5 in depth.

6.1 What is Classification?

Classification is the central operation for the recognition of patterns, i.e. the attempt to give a designation to observed phenomena in order to move from the purely descriptive level of physical, geometric or other features to a symbolic level. Within the scope of pattern recognition in technical applications we can define classification as the assignment of an object characterized by a set of features to one of a number of predefined classes.

This definition is somewhat restrictive. It is quite possible that the classes will not be previously known but created over time on the basis of similarity relationships. Nevertheless, the definition corresponds to the typical structure of industrial classification problems. It is perhaps best understood by taking a look at some examples for classification tasks listed in Table 6.1.

This list of applications shows that there are very different types of classification problems even when we restrict ourselves to technical applications. All these examples have one thing in common though: moving from raw data to features and class assignment, the amount of data is significantly reduced. For example, the image of a character should have at least 400 pixels in order to be

Table 6.1 Examples for technical classification problems

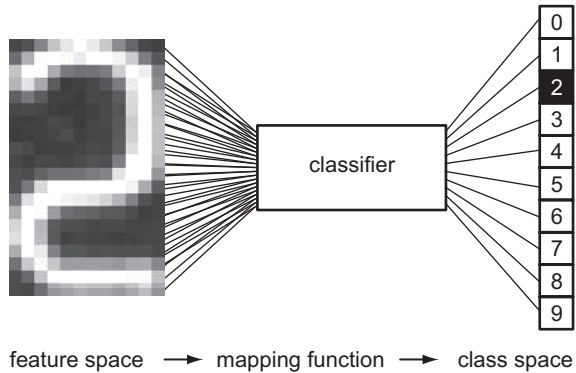
Application	Features	Class description
Distinguishing objects (e.g. screws)	• Image of the object • Dimensions of the object • Shape parameters (ratio of perimeter and area, structure width,...)	Designation of the object
	...	
Distinguishing surface defects	• Image of the defect • Dimensions of the defect • Brightness properties of the defect	Designation of the type of defect (scratch, blowhole...) and/or rating of the defect as still permissible or not
	...	
Character recognition	• Image of the character • Fourier descriptors of the character • Shape properties of the character (like distances, curvatures,...)	Name of the character or number of the character in a given character set
	...	
Noise test of an electrical motor	• Time waveform of the noise signal • Spectral properties of the noise signal	Rating of the motor as good or faulty
	...	

identified reliably; the spectral properties of acoustic signals are typically characterized by at least 256 spectral lines. Each descriptive component is a feature. Every combination of features makes up a specific object. The number of classes, on the other hand, is typically much smaller, so that classification can be seen as a kind of data compression. This rule is of course not without exceptions: screws, for example, can be differentiated in several classes according to a single feature, their diameter. However, one would typically not use one of the powerful classifiers we are going to introduce below for such a task. We can therefore say that considerable step-wise data reduction is typical for image processing applications that involve actual classifiers, from the input image to feature values and finally to the class assignment.

Above we defined classification as the assignment of an object, described by a set of features, to a specific class. This can also be formulated as a mapping from a feature space to a class space. The set of features can be written as a vector of values, just as the set of classes can be written as a vector of class numbers. In mathematical terms, classification then becomes a transformation between two vector spaces which can be performed by a mathematical algorithm. Figure 6.1 illustrates this using the example of digit recognition.

In the following we will have to introduce some theoretical foundations without going too deeply into mathematical detail. Then we will present some types of classifiers and discuss one of these types, a specific neural network, in detail.

Fig. 6.1 Classification as mapping between vector spaces



6.2 Classification as Function Approximation

This section summarizes some theoretical foundations from the field of statistical pattern recognition. As we have seen in the previous section, classification can be viewed as a mapping from a feature space to a class space, i.e. as a transformation function. This function is typically unknown since we do not know how a human determines which character is depicted from the image of a character. In order to automate this classification process using a computer, we will have to approximate this unknown function using numerical methods.

This immediately raises the question which function is to be approximated. The answer depends on the given problem, but in image processing applications, the typical task is to imitate an assignment prescribed by a human inspector. For example, we expect a character recognition system to assign the same character designation to an image as a human observer would do. Likewise, we expect a classification system for surface defects to reject large scratches as would a human inspector. The task of the classifier then consists in performing the assignment between input features and class designations as a human being would do. But this does not tell us how we can derive a classification function that can be calculated by a computer. Before we can attempt to solve this question, we have to introduce some basic terms.

6.2.1 Basic Terms

There are many technical applications which allow for easy quantification of permissible feature ranges; for example, allowances given on a technical drawing or definitions for surface quality can be checked in a comparatively simple way. However, many of the ratings and classifications naturally performed by humans cannot be easily formalized, not least because humans in general do not know or cannot describe the criteria they use when performing these classifications.

How would you describe the difference between the letters “F” and “E” in such a way that a machine can understand it? Simple, you might say, the “F” has one stroke less than the “E”, but this does not solve anything. Now you have to explain to the computer that the abrupt 90° change of direction at the bottom left corner of the “E” is the beginning of a new stroke because after segmentation the character is simply one contiguous object, not a collection of distinct strokes. Humans usually derive the mappings they use for this kind of classification from examples. If you see an “E” and an “F” often enough, you will be able to distinguish between the two—and will naturally conclude that one of the letters differs from the other by one stroke. Should it not be possible to perform this learning process, which derives the necessary mapping from a set of examples, automatically in a machine?

This is indeed feasible and it is called *machine learning*. This kind of learning is based on a collection of samples for which the correct (read: prescribed by a human) correspondence between object description and class assignment is known. The automatic classifier “learns” to imitate these assignments and to apply them to new objects it has not encountered during learning. Let us first introduce some terms:

Pattern: Denotes the description of an object to be classified. Patterns can be represented in very different ways depending on the application, e.g. it can be a normalized image of a character, but also a set of comparatively abstract measurement values.

Feature: Denotes a single value within a pattern. If an object is described by an image, the brightness of a specific point in this image is a feature. Area, perimeter, and average curvature are other examples of features.

Feature vector: All features describing an object are collected in a feature vector. In this sense, the feature vector is equivalent to the *pattern*. Since the feature vector forms the input to the classifier, it is denoted by i for *input*. A pattern is then represented by the vector of all feature values:

$$i = \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_N \end{pmatrix}. \quad (6.1)$$

Class index: The class index is the number of the class that an input pattern has been assigned to—be it by the human acting as the instructor or by the classifier itself.

Target vector: The target vector is a vector of all zeros, with one exception: the element corresponding to the correct class index of the input pattern has a value of 1. The target vector is denoted by t for *target*. It looks as follows:

$$t = (0, \dots, 0, 1, 0, \dots, 0), \quad (6.2)$$

where the element having the value 1 is the element t_k , i.e. the element corresponding to the class index.

Output vector: The output vector contains the result of the classification process. Most classifiers give (at least internally) a separate output value for each of the possible classes. Each output value indicates the probability that the pattern belongs to the corresponding class. Ideally, the output vector is identical to the target vector; in practice, it will always be different to some extent. We will denote the output vector by o for *output*.

Training set: The training set is a collection of patterns for which the correct target vectors (or class indices) are known. The training set thus describes the task to be learned by the classifier.

Using these terms, a classification is a mapping from a space of (analog) features into a space of (discrete) classes. We place the properties “analog” and “discrete” in parentheses because they denote rather the typical way to look at such problems than the actual mathematical form of the values. Discrete features are actually used quite frequently in digital image processing since the gray levels of a digital image are themselves discrete. On the other hand, classifiers can also yield analog output values. Typically, however, the features are treated as real numbers, even when they are restricted to specific values, i.e. they are simply used in arithmetic operations. In contrast, the results are in the end represented as a class index that can take only discrete values, although the classifier itself computed the probabilities for each class as real numbers.

6.2.2 Statistical Foundations

In the previous section we explained that the result computed by a classifier indicates the probability that the pattern belongs to one of the given classes. We also used some other statistical terms. It appears that classification and pattern recognition are somehow connected to mathematical statistics. Why is that?

As we have seen above, it is usually not possible to write down an analytical expression for the mapping from feature space to class space. Classification tasks are therefore approached by trying to derive an approximation for this mapping from examples. The examples, however, represent only a section of the problem. If the task is to build a classifier which identifies a character from an image of only 200 pixels, then there are 256^{200} possible combinations of gray levels in the input image. This allows for more than $4 \cdot 10^{481}$ distinct input patterns, far more than there are molecules in the universe. Of course, most of these patterns are very, very similar to each other and the overwhelming majority will not resemble any character at all. Nevertheless, it is obviously impossible to represent more than a very small sample of this problem in the training set. This is even more true of problems with analog feature values where the number of possible combinations is infinite, not only in a practical sense but also in actual mathematical terms.

We therefore have to take a sample from the problem and draw conclusions about the behavior of the entire system from this essentially random sample. This inevitably leads to uncertainties since the sample used for training cannot describe

the problem in its entirety. Of course, one will try to compile the training set in such a way that the problem is described as comprehensively and precisely as possible. Nevertheless, the classifier will be forced to extrapolate when confronted with new patterns, i.e. to use the “knowledge” gained from the training set to infer the probability with which the new pattern belongs to one of the known classes.

The following discourse is not intended as an introduction to statistical decision theory. This would go far beyond the scope of this text. Instead we want to give a feel for the properties of machine learning and classification. Therefore, we use a rather qualitative approach, leaving out most mathematical details.

The training set is a sample of the overall problem. We can regard the problem itself as a pattern-generating process. An arbitrary combination of feature vector and target vector will occur in this process with a probability depending on both vectors, i.e. their joint probability distribution¹ $P(i, t)$. Many of the values of this probability distribution, in fact most of them, will be 0.

The objective of the classification process is to decide from a feature vector i which class the pattern belongs to, i.e. which output vector o corresponds to the given feature vector. The basis for this decision is the *conditional probability* $P(o|i)$, which gives the probability for the presence of the class represented by o under the condition that the input vector i is present. This probability is also called *posterior* or *inference probability* since the classifier tries to deduce the class from the feature vector. The relationship between the conditional probability and the joint probability is given by Bayes’ theorem as follows:

$$P(i,o) = P(o | i) \cdot P(i). \quad (6.3)$$

Spelled out, this means that the probability for an arbitrary combination of feature vector and target vector is given by the product of the probability that the feature vector i occurs at all and the probability that—under this condition—class o is present.

6.2.3 Defining Classifiers

Someone defining a classifier has to ascertain that the classifier performs the desired mapping from feature space to class space in the best possible way. The mapping itself, as we have seen, is given by the training set. It remains to be seen what “as well as possible” means in this context.

The classifier can be regarded as a decision function $d(i)$ computing an estimate $\hat{o}$ of the true output vector o from the feature vector i :

$$d(i) = \hat{o}. \quad (6.4)$$

¹ For analog values it would actually be correct to speak of probability densities, but the basic argumentation stays the same.

Actually, this decision function d is a vector of k functions, one function for each class. For a given input vector, a vector is computed whose values d_k give the probabilities for the input vector belonging to class k . Usually the pattern will be assigned to the class for which the maximum probability value has been computed.

The estimate $\hat{o}$ will inevitably be different from the true value o . The objective for the construction of a classifier is therefore to minimize this difference. This in turn requires the measurement and evaluation of the difference in the first place. This is done using a cost function. Depending on the cost function used, very different classifiers can be defined. We will mention here only one, which is of great theoretical importance.

The Bayes classifier: An obvious cost function assigns a certain cost c_f to all classification errors and another cost c_r to all rejections, i.e. cases where no classification was deemed possible. A correct classification does not create any cost. The resulting classifier is called *Bayes classifier*. It has the important property of being the classifier with *minimum probability of error*. Its main problem is that it is in general not possible to construct this classifier because it requires complete knowledge of the statistical properties of the entire problem domain, i.e. the probability distributions mentioned in the previous sections have to be known.

Such comprehensive information is a luxury practically never enjoyed in practice, therefore we have to use approximation methods. One possible method is to assume certain probability distributions. For example, one can assume that feature vectors are governed within each class by an N -dimensional Gaussian density (with N as the number of features, i.e. the dimension of the feature space). Unfortunately, the resulting classifier is hard to handle because of the high-dimensional feature spaces typical for image processing. Furthermore, the basic assumption cannot be verified in practice.

Practical procedure: In practice there are two ways to construct a usable classifier, leading to two different basic types of classifiers:

1. A number of typical patterns, so-called *prototypes*, are stored for each class and compared to the observed feature vectors. A feature vector is assigned to a class according to its similarity to the prototypes. These classifiers can be characterized by the method used to determine the prototypes and by the computation of the similarity measure. Such classifiers are called *instance-based classifiers* (cf. Sect. 6.3).
2. The mapping function between feature space and class space is assumed to have a certain functional form. Parameters are then adjusted so that an optimality criterion is minimized for all patterns from the training set. Typically the mean square deviation between the output vector o of the classifier and the target vector t is used as the optimality criterion:

$$E(|t - o|^2) \stackrel{!}{=} \min. \quad (6.5)$$

Classifiers using a functional approach differ mainly in terms of the type of functions and the optimization method used. Polynomial classifiers are one example for this type of classifier, artificial neural networks, discussed in Sect. 6.4.2, another.

6.3 Instance-Based Classifiers

Instance-based classifiers use a number of prototype patterns for the individual classes and assign an observed input pattern to the class with the prototype pattern most similar to the input pattern. In a sense, they take “learning from examples” quite literally: the classifier is actually defined by the examples. The various types of instance-based classifiers differ in their methods of selecting the prototype patterns and of computing the similarity between prototype and input pattern.

6.3.1 Nearest Neighbor Classifier

The most basic instance-based classifier is the nearest neighbor classifier. It uses simply the entire training set as its reference set. Every pattern from the training set thus becomes a prototype pattern, denoted by p_i . The usual similarity measure is the Euclidean distance. The distance d_i between the input vector i and a prototype pattern p_j is:

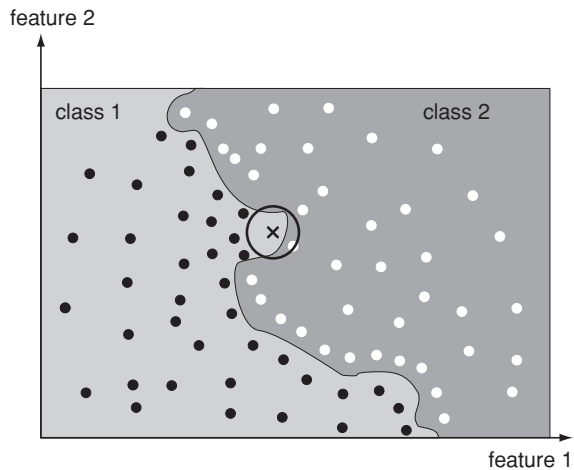
$$d_j = \|i - p_j\| = \sqrt{\sum_{k=0}^N (i_k - p_{jk})^2} \quad (6.6)$$

with i_k as the individual feature values of the input vector i , p_{jk} as the feature values of the prototype patterns p_j . From the point of view of functional approximation, the classification function is approximated by using all those points where it is precisely known, i.e. the prototype patterns with their known class assignments.

Every input pattern is compared to all prototype patterns in this way. It is then assigned to the class of the closest prototype, i.e. the prototype for which the Euclidean distance in Eq. (6.6) becomes minimal. We can already see the main problem of the nearest neighbor classifier: increasing the size of the training set will of course increase the knowledge the classifier has about the problem; at the same time, however, the amount of computation required for the classification of an input pattern increases proportionally since the input pattern has to be compared to all available prototypes.

It can be shown that for an unlimited training set, the nearest neighbor classifier reaches an error rate less than double that of the Bayes classifier, the theoretically best possible classifier. Of course, unlimited training sets are even theoretically impossible, but using a nearest neighbor classifier becomes impractical much

Fig. 6.2 Approximation of complex class boundaries by a nearest neighbor classifier



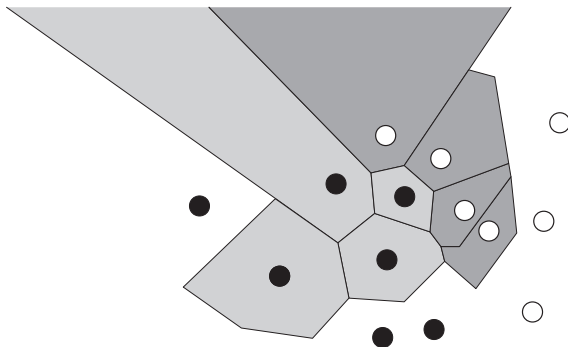
earlier due to the computation effort for the comparison with all prototypes. Especially the approximation of complex, interlocking class boundaries by individual prototypes is very difficult. Figure 6.2 illustrates this for a classification using two features. The prototypes of class 1 are drawn as black circles, those of class 2 as white circles. The black cross represents an input pattern of class 1. It is close to the class boundary and the closest prototype actually belongs to class 2, indicated by the dark circle around the pattern. Obviously, a high density of prototypes is required close to complicated class boundaries in order to achieve reliable classification.

Border problems: Each prototype of a nearest neighbor classifier defines a certain area within the feature space. The size of this area depends on the distances between the prototype and its neighbors, the borders being the perpendicular bisector on the lines connecting the prototypes. Every pattern inside this area around a prototype will be assigned to the class of the prototype. A problem arises towards the borders of the feature space: there the areas stretch into infinity where they will most probably include combinations of feature values not corresponding to any useful pattern. Figure 6.3 uses a section from Fig. 6.2 to illustrate this problem: where there are still neighboring prototypes, the areas have meaningful borders; towards the upper edge, though, they continue into infinity. This can lead to rather arbitrary classification results for input patterns outside the range covered by prototypes.

6.3.2 RCE Networks

The border problem of the nearest neighbor classifier depicted in Fig. 6.3 can be avoided by assigning a sphere of influence to each prototype. An input pattern will

Fig. 6.3 Border problem of nearest neighbor classifiers



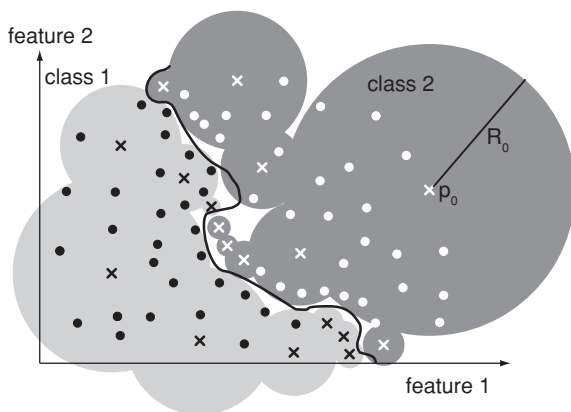
then be assigned to a class only if its distance from the nearest prototype p_i is at most R_i . Figure 6.4 shows that this amounts to an approximation of the class distribution by means of overlapping circles in a two-dimensional feature space. When more than two features are used, these circles become spheres or hyperspheres. A welcome side-effect is that the number of prototypes can be reduced considerably in comparison to a nearest neighbor classifier: only the circle centers indicated by crosses are required as prototypes.

This classifier model will assign input pattern i to the class of the nearest prototype p_j , provided that

$$d_j = \|i - p_j\| = \sqrt{\sum_{k=0}^N (i_k - p_{jk})^2} \leq R_j. \quad (6.7)$$

This can be described as a *nearest neighbor classifier with rejection threshold*: an input pattern which is not sufficiently close to any prototype will not be classified at all; it is rejected. This type of classifier is often referred to by the name of a neural network model operating according to this principle. In analogy to the law governing the decrease of electrical field strength with the squared distance, these

Fig. 6.4 Nearest neighbor classifier with rejection threshold/RCE network



networks are called *Restricted Coulomb Energy (RCE) networks* (Reilly et al. 1982). RCE networks are not susceptible to the border problem of the nearest neighbor classifier since an input pattern far outside the area covered by the training set prototypes will not be sufficiently close to any prototype to be classified at all. It is still true, however, that the approximation of complex class boundaries requires a high density of prototypes.

6.3.3 Vector Quantization

In connection with RCE networks we have seen that under the appropriate circumstances the number of prototypes can be drastically reduced in comparison to that of a nearest neighbor classifier. The RCE network uses existing prototype patterns as centers of circles (or of hyperspheres for higher-dimensional problems). The radii of these circles were then adjusted for optimum coverage of the area of a particular class. It is also possible to change the position of the center points so that they no longer correspond to existing training patterns, but instead act as a kind of “average pattern” which represents the respective class—or part of it—in the best possible way.

The construction of such a classifier can be described as follows: Find a set of feature vectors q_j such that the error caused by replacing every vector i_k by the nearest vector q_j is minimized. Depending on the concrete procedure, the number of representative feature vectors can be given a priori or determined dynamically. Such methods are called *vector quantization* because the entire set of all possible feature vectors is represented by the discrete, i.e. quantized, set of centroid vectors. Vector quantization is covered in depth, for example, in (Kohonen 1989).

6.3.4 Template Matching

Template matching is usually discussed in the context of segmentation instead of classification, as we have done in Sect. 4.5. However, the mechanism of template matching is identical to that of a nearest neighbor classifier with an additional rejection threshold: every image section is compared to all existing templates and assigned to the class of the template most similar. There are two main differences to the standard nearest neighbor classifier:

Similarity measure: Instead of the Euclidean distance from Eq. (6.6), the correlation between template and image section according to Eq. (4.1) is used.

Rejection threshold: Akin to the minimum similarity required for template matching when it is used as a segmentation method, a minimum correlation value is required in order for a class assignment to be performed at all.

Obviously, template matching can be used in the same way as a nearest neighbor classifier. The similarity extends to the computation requirements for large numbers of prototypes and the behavior at complex class boundaries. We will see in Sect. 9.3 that template matching is not a very selective classifier due to the properties of the correlation algorithms. The method is therefore suitable only for clearly distinguishable patterns of rather high quality.

6.3.5 Comments on Instance-Based Classifiers

The previous sections have presented several types of classifiers, all of them using prototype patterns. Each prototype pattern rules a part of the feature space defined by a distance measure and possibly a rejection threshold. These classifiers can therefore be characterized as *integrating* classifiers because each prototype integrates a certain region of the feature space into its class.

Another common property of the various instance-based classifiers is that they can, in general, be constructed and trained quite rapidly. This is most obvious in the case of the nearest neighbor classifier and the template matching algorithm—which actually function identically, apart from the similarity measure.

Classifiers of this kind are not well suited to high-dimensional and strongly non-linear problems because they need an extremely large number of prototypes for complex class boundaries, which leads to impractical computation requirements. Their main disadvantage, however, is that they do not perform any weighting of the features. For example, if a nearest neighbor classifier uses a resampled image as its feature vector, it will assign the same importance to every pixel since all pixels contribute equally to the Euclidean distance between input pattern and prototype. Differences in unimportant areas of the pattern will therefore be treated identically to differences in significant areas. A little background noise can add up to the same distance values as the complete absence of a stroke in a character. Small changes in the input pattern can therefore lead to considerable shifts of the classification results.

This problem can, of course, be reduced by introducing appropriate weightings and feature assessments. However, this requires considerable effort because it must first be determined which features are significant in the context of the application and the training set. In contrast, the function-based classifiers we will discuss in the following section are much more robust, mainly because of the optimization approaches used for training.

6.4 Function-Based Classifiers

The second basic type of classifiers tries to approximate the decision function directly instead of deriving prototypes from the training patterns. The main difference between the various types of function-based classifiers lies in the family of

functions used for the approximation. Commonly used function families include polynomials, forming the basis of the polynomial classifier, and exponential functions, which are typical for many types of neural networks.

6.4.1 Polynomial Classifier

The polynomial classifier, discussed in detail in (Schürmann 1977) approximates the individual functions d_k of the decision vector d by polynomials in the elements of the input vectors i :

$$d_k = a_{0,k} + a_{1,k}i_1 + a_{2,k}i_2 + \cdots + a_{N,k}i_N + a_{N+1,k}i_1^2 + a_{N+2,k}i_1i_2 + \cdots \quad (6.8)$$

Combining all terms $i_1^a i_2^b \dots i_N^c$ of the polynomials in a vector x , the complete approximation for the vector of decision functions can be written in compact matrix form as:

$$d = Ax. \quad (6.9)$$

Typically, some restrictions are made on the polynomial terms. A simple regular approach uses all terms up to a degree of G . The coefficient matrix A can then be optimized using a least squares method, i.e. the sum of squares of the differences from the true target vectors is minimized:

$$\sum |o - Ax(i)|^2 \stackrel{!}{=} \min. \quad (6.10)$$

The required condition for a minimum is that the partial derivatives of the above sum of errors—with respect to all polynomial coefficients—vanish. This leads to a system of linear equations for the coefficients in A , which can be solved using common algorithms for systems of linear equations, which we won't explain any further.

Approximation properties: The approximation theorem by Weierstrass (Bronstein et al. 2005), states that any continuous function can be approximated to arbitrary accuracy using a polynomial of sufficiently high order. Polynomials are therefore a dense functional basis for approximation and the polynomial classifier is a universal approximation tool. However, the number of terms in a complete model will increase so fast as to be nearly impossible to optimize. This problem can be lessened, for example by using only a selection of polynomial terms of higher order employing application-dependent selection rules. Furthermore, the polynomial classifier can be generalized to a functional-link classifier according to Pao (1989), by using not only polynomial terms but arbitrary functions of the input vector. This does not constitute a fundamental change since the new functions could also be approximated by polynomials, but it can reduce the number of free parameters considerably.

6.4.2 Multilayer Perceptron-Type Neural Networks

The concept of neural networks was derived from the idea of obtaining certain processing results, especially in the area of pattern recognition, in analogy to biological brains by interconnecting a multitude of simple processing units in a complex network. For quite some time a strong antagonism existed between advocates of “classical” statistical methods and the protagonists of neurocomputing. If we disregard for a moment the more biologically and psychologically motivated branches of neurocomputing, which aim toward modeling actual processes in the nervous system, neural networks are nowadays regarded mostly as a new way of looking at statistical modeling, i.e. as a specific way of representing and implementing statistical methods. It is indeed possible to implement the classifiers presented so far in terms of neural networks, as we already mentioned in connection with the RCE model.

The new approach introduced in the form of neural networks has produced a number of interesting results. An important difference between classical methods and those of neurocomputing is that the latter typically use more input variables and apply non-linear function bases.

This is not the right place to give a comprehensive overview of neural networks in general. Instead we will give a practically oriented introduction into the large and varied research area of neurocomputing using the most common type of neural network in technical applications, the multilayer perceptron.

Structure and processing units: The building blocks of artificial neural networks are often simply called *neurons*. Because of the marked differences in the capabilities of biological and artificial “neurons”, we will use the term *processing unit* or, as is common in the literature, simply *unit*.

Figure 6.5 shows the typical structure of a multi-layer perceptron. It usually consists of three layers of processing units, one input layer, one so-called hidden layer, and one output layer. Between those layers of units there are two layers of interconnections. The units of the input layer are indicated by i , those of the hidden layer by h , and those of the output layer by o . There are several variations of this structure, for example there can be direct connections between the input and the output layer, called *shortcut connections*. More than three layers are also possible, but the structure depicted here is the most common: three-layered² and strictly hierarchical, i.e. there are no connections between layers which are not directly adjacent.

Figure 6.6 shows a block diagram of a single processing unit. Each processing unit is equipped with one or more inputs. Each input is characterized by a

² There is some disagreement in the literature about the nomenclature; some authors call this a three-layered network since there are three layers of processing units, others speak of two-layered networks, arguing that the input layer only collects the input values and performs no processing itself.

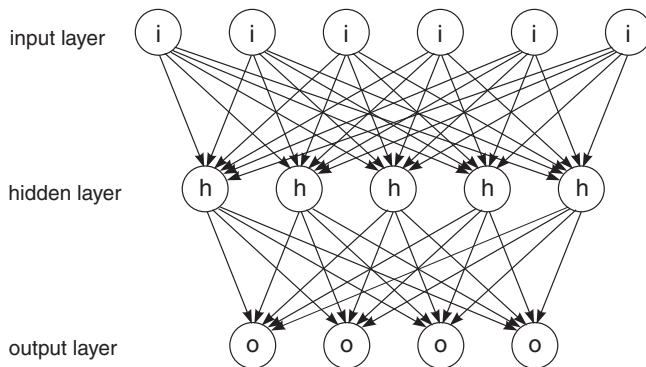


Fig. 6.5 Typical structure of a neural network

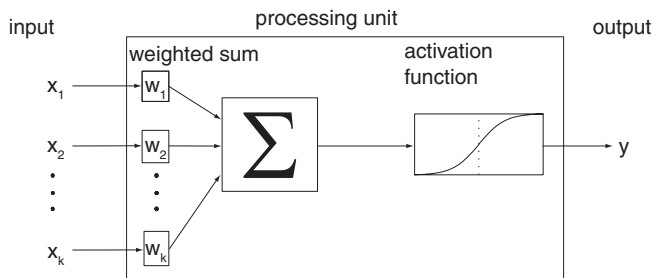


Fig. 6.6 Block diagram of a processing unit in a multi-layer perceptron

weighting factor or simply weight w_k . The output value of the unit is computed from the weighted sum of the input values.

The activation state or simply activation of a unit j is a function of the weighted sum of the outputs o_k of all units k connected by weighted connections w_{kj} ³ with unit j :

$$a_j = f_a \left(\sum_k w_{kj} o_k - \theta_j \right). \quad (6.11)$$

The multi-layer perceptron uses almost exclusively sigmoidal functions as activation function. Figure 6.7 shows the two most common activation functions for multi-layer perceptron units.

The *logistic function* is used most often.

$$f_a(\text{net}_j) = \frac{1}{1 + e^{-(\text{net}_j - \theta)}}. \quad (6.12)$$

³ Again various notational conventions are used in the literature; we will always name the unit from which the connection goes out first, the receiving unit last.

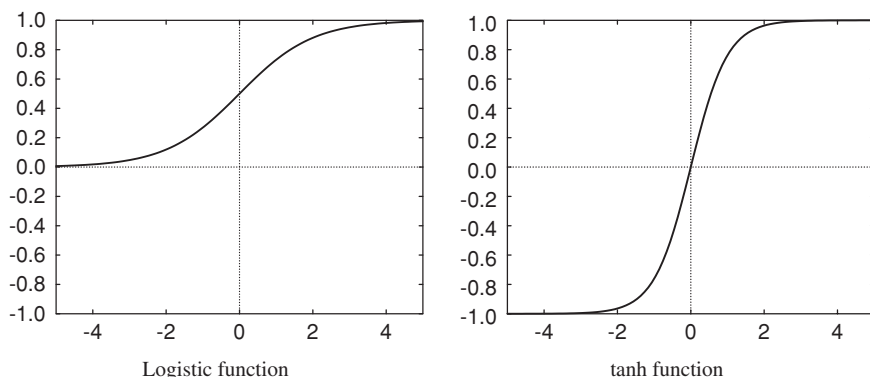


Fig. 6.7 Sigmoidal activation functions

Because of their asymptotic behavior both functions automatically put a limit on the activation value independent of the size of the input sum. In some cases it is advantageous to have a bipolar activation state so the tanh-function is used, which is quite similar to the logistic function except that it takes values between -1 and 1 instead of 0 and 1 .

Propagation and adaptation: Further characteristics of neural networks are the *propagation strategy* and the *adaptation rule*. The propagation strategy determines the way in which activation values are propagated through the network. The multi-layer perceptron is a *feed-forward network* which means that the activation is propagated layer by layer from the input towards the output without any feedback connections.

The adaptation rule determines how knowledge is incorporated into the network. In multi-layer perceptron networks this is done by changing the weights between the units in a *supervised learning process*. During each training cycle all patterns from the training set are classified by the network. The results of the network are compared to the target vectors in the training set. Then the weights are adapted so that after a number of training cycles the network is capable of classifying the training patterns with a predefined accuracy. This process is called supervised learning because the current output of the network is supervised by a comparison of actual and desired output, the result of which is used to drive the direction of the training.

Most supervised learning methods operate similarly to the optimization of the coefficient matrix of the polynomial classifier by minimizing the sum of squared errors at the outputs of the network. The most popular training method is the *generalized delta-rule*, commonly known as *error backpropagation*. This algorithm changes the weights in each training cycle in the direction of decreasing error. It is explained in detail in Sect. A.1.

Notwithstanding some drawbacks of the algorithm, multi-layer perceptron networks trained by backpropagation have proved themselves to be powerful and robust classifiers for a wide range of applications. It should be mentioned that, like

polynomials, exponential functions such as the logistic function facilitate the construction of universal approximators, i.e. a neural network of sufficient size is capable of approximating any function to any desired degree of accuracy.

6.5 Comments on the Application of Neural Networks

This section could as well have been called *rules for using statistical classification methods* because some fundamental rules have to be observed with every statistical modeling and classification method. This section will present the most important rules for the practical application of such methods.

6.5.1 Composition of the Training Set

The training set has to be a sufficiently exact and comprehensive sample of the entire problem domain. It will therefore have to contain “borderline patterns”, i.e. patterns which are rated as barely tolerable or just still readable. It is also important to represent all classes with approximately equal numbers of patterns. An underrepresented class contributes less to the error function optimized during training and will therefore not be learned as well. This effect can be reduced to a certain extent by repeating patterns from under-represented classes more often in each training cycle, preferably altering them slightly by adding noise. One has to be aware, however, that this does not introduce additional information about the problem into the training process. Actual training patterns are always preferable to artificial duplication.

6.5.2 Feature Scaling

Comparable value ranges of the individual features are very important for any statistical method to prevent singular features from dominating the others. Incorrect scaling can also cause problems during training and reduced selectivity of the classifier. It can be observed in neural networks, for example, that for absolute values of the input signals of 5 and up, the logistic activation function will reach a value of almost 1 and be nearly flat. This effect is called saturation and has the following consequences:

- The network can no longer differentiate between different input values, it loses its selectivity.
- Training becomes extremely slow because most training algorithms use derivatives of the activation function, which are nearly zero in the flat areas.

A common scaling method is a normalization of the input values to the interval $[-0.5; 0.5]$. Its advantage is that the activation function is at its steepest in this area, i.e. its derivative is at its maximum, making training easier. For input signals whose possible value range is well known beforehand this scaling method is very simple to perform and yields good results. The individual pixels of a resampled image are a typical example.

The ranges of other features, like object dimensions or areas, cannot be determined beforehand. Sometimes very large values can occur here. Scaling such features to a fixed minimum and maximum risks compressing small values so much that they are almost indistinguishable. In such a case, findings from multivariate statistics suggest performing a standardization over the entire training set. This means that mean and standard deviation are computed for each feature over all patterns of the training set. Afterwards the values are scaled such that the mean value becomes 0 and the standard deviation 1. The majority of values will then be densely distributed around 0 but the variation of the values is still recognizable.

6.5.3 Rejection

Multi-layer perceptron networks and polynomial classifiers are prone to the same problem as the nearest neighbor classifier: for input patterns outside their “domain of experience”, i.e. far from all patterns contained in the training set, they can yield arbitrary results. The reason is that these classifiers operate by dissecting the feature space by a number of hyperplanes separating the individual classes. They can therefore be called *separating* classifiers, in contrast to the *integrating* instance-based classifiers, which integrate similar patterns in the sphere of influence of a prototype. The discriminating hyperplanes are created in a completely different way than the class boundaries of the nearest neighbor classifiers, but the effect is identical: every pattern on one side of the discrimination hyperplane is assigned to one particular class. Since the hyperplanes stretch into infinity, the classifier will handle nonsensical patterns without thinking twice about it.

This is a general problem of statistical classification methods. The approximated class probability densities are usually not defined over the entire feature space. This means that there are large areas where no training patterns are possible at all. The approximation functions, however, are by nature defined over the entire feature space.

Extensive research has been performed on the problem of rejection schemes for perceptron type networks. They reach from simple plausibility checks to complete statistical models of the space covered by the training patterns. A very useful rejection rule for well-trained classifiers uses the contrast between the class deemed most probable and the runner-up.

The class assignment is derived from the output unit exhibiting the highest activation value. With o_{max} as the highest values, o_{max2} as the second highest value in the output layer, the *classification certainty* can be defined as:

$$g = \frac{o_{max} - o_{max2}}{o_{max} + o_{max2}}. \quad (6.13)$$

The larger the difference between the two values, the closer to 1 the certainty value will be. The certainty value will be 0 when both activation values are identical, i.e. when the network is completely uncertain which class the pattern belongs to. It has to be noted however, that this certainty measure will not give meaningful results when the classifier is confronted with input patterns unrelated to the task for which the classifier has been trained. This is especially true if the input pattern belongs to a class which is unknown to the classifier, because for an unknown class no discrimination hyperplane and thus no decision function exists. The classifier is then forced to assign the pattern to one of the known classes. Since the classifier is totally ignorant of the fact that there can be other classes than those it has encountered during training, it may exhibit no uncertainty at all about this decision. Experience shows that this kind of “silent misclassification” will not occur in a typical technical application where a classifier has been trained for a larger number of previously known classes.

For further discussion of this topic we refer to the extensive literature on neural networks, for example (Zell 1994, Bishop 1995, Smith 1993 and Brause 1991).

6.5.4 Differentiation from Other Classifiers

At the end of this chapter we want to comment on an objection sometimes raised against neural networks. It is sometimes said the results of neural networks are unpredictable, arbitrary, random, or subjectively biased. Such claims are made especially by advocates of classical methods, like correlation, and are of course of some importance for anyone who has to design a quality control system.

The claim of unpredictability is obviously unfounded for the classification process itself; after all, its result is computed and it is computed in exactly the same way every time and will always yield the same result for the same input pattern. More cannot be said of any method. In this respect, neural networks do not differ at all from “classical” methods. It is therefore useless to try to construct a conflict around this. It is true, however, that there is no particular place where the network stores its knowledge about a particular input pattern. In an instance-based classifier, on the other hand, it is possible to determine which prototype led to a certain result. Therefore, the behavior of an instance-based classifier may appear more transparent, more easily understandable.

The objection of arbitrariness can therefore relate only to the training process. Neural networks are usually initialized with random numbers; several training runs with the same training set will therefore not result in identical classifiers. These differences are insignificant for the behavior of well-trained classifiers with a sufficiently low and stable residual error.

Closely connected to the claimed arbitrariness is the objection concerning subjective bias. It relates to the fact that neural networks are adaptive systems that learn from training samples selected, compiled, and pre-classified by humans. Again, however, neural networks do not differ from other statistical methods, which will also arrive at different results for different samples. This is even true for simple methods like correlation, which, like the nearest neighbor classifier, is based on a comparison of the current pattern with a number of stored patterns. These patterns also have to be selected and assigned to classes at some time before putting the classifier into operation. Again neural networks are no different in this respect.

One of the main reasons for these objections to neural networks is probably the fundamental difference in the operating method of the two basic types of classifiers. We mentioned in connection with instance-based classifiers that their main disadvantage is the lack of weighting of the features. This makes them very noise-sensitive. Neural networks are much more robust, but it is often difficult to understand how they derive the weightings for the different features during training.

6.6 Summary

Classification: In technical applications we define classification as the assignment of an object characterized by a set of features to one of a number of predefined classes.

Classification as mapping: Classification can be described as a mapping from a feature space to a class space. For most classification applications, like character recognition, no formal transfer function is known for this mapping.

Pattern: In a classification context, pattern denotes an object described by a set of features; equivalent to feature vector.

Machine learning: Field of research concerned with the construction of adaptive systems, especially the derivation of a classification function from examples of a predefined assignment (possibly done by a human) between patterns and classes.

Feature: Single value describing one aspect of an object, e.g. a dimension or the brightness of a pixel in a resampled image.

Feature vector: This vector contains all feature values describing an object.

Target vector: The target vector in machine learning contains the correct, predefined class affiliation.

Output vector: The output vector contains the probabilities for the class affiliation of a pattern determined by the classifier.

Training set/training data set: Collection of patterns with assigned class information used to adapt a classifier to the problem to be learned.

Supervised learning: Denotes training methods which perform a comparison of the actual and desired output of the classification result after each training step and correct the parameters of the classifier in accordance with the comparison

result. In contrast, a classifier trained by *unsupervised learning* will derive the class structure on its own based on similarities between the training patterns.

Border problem: The border problem is caused by the fact that the training set cannot cover the entire feature space. Confronted with an input pattern far outside the area represented by the training set, a classifier can yield arbitrary results because it has no basis for fitting this pattern into its class model.

Bayes classifier: The Bayes classifier has the theoretically minimal rate of error. Since it requires full knowledge of the statistical properties of the problem, it is usually impossible to construct in practice.

Instance-based classifier: This type of classifier determines the class affiliation of a pattern according to its similarity to stored prototypes. Such classifiers can be constructed and trained very easily. Their main disadvantages are the large number of prototypes required to approximate complicated class boundaries and the computation time required for the comparison of a pattern with all prototypes.

Nearest neighbor classifier: The most basic type of instance-based classifiers; it determines the similarity of a pattern to the prototypes using the Euclidean distance.

RCE network: This variation of the nearest neighbor classifier uses a rejection threshold for each prototype. Patterns farther away from the closest prototype than its rejection threshold will not be classified. The area of each class is thus approximated by circles (or spheres and hyperspheres for higher-dimensional problems). This technique avoids the border problem of the nearest neighbor classifier and reduces the number of required prototypes.

Vector quantization: Denotes the representation of the domain of feature vectors by a limited number of prototype vectors. This technique can be used to reduce the number of prototypes for a nearest neighbor classifier or as a classification method of its own.

Template matching: This type of nearest neighbor classifier uses the correlation coefficient instead of the Euclidean distance to evaluate the similarity between input patterns and prototypes.

Function-based classifier: This basic type of classifier uses a function approximation to model the decision functions for the class assignment directly. Subtypes differ in the function family used as basis for the approximation. Their advantages are high processing speed in the classification phase and the ability to represent very complex problems. Disadvantages are border problems and possibly complicated and computation intensive training methods.

Polynomial classifier: This type of function-based classifier uses polynomials to approximate the decision functions. Polynomials of sufficiently high order are a complete functional basis, i.e. they can approximate any function to any desired accuracy. The computation of the coefficients of the classifier can become very complicated with increasing order of the polynomials. The coefficients are determined by minimizing the square distance between the output vectors of the classifier and the target vectors of the training set.

Multi-layer perceptron network: This common type of neural network is a function-based classifier based on exponential functions, which are, like

polynomials, a universal approximation system. They are usually trained using the backpropagation algorithm.

Backpropagation: Most common training method for neural networks. It is based on gradual changes of the internal weights of the network in the direction of decreasing error. The error criterion is the mean square distance between the output vectors of the network and the target vectors of the training set.

Feature scaling: A prerequisite for successful application of statistical modeling and classification methods is an appropriate transformation of the features to comparable scales in a range where the classification functions are sufficiently selective. For many methods from multivariate statistics as well as for neural networks, a standardization to mean value 0 and standard deviation 1 yields good results.

Rejection: To avoid the border problem of neural networks and other classifiers, rejection rules can be applied. Unexpected results for input patterns outside the specified and trained problem domain cannot be avoided entirely, however.

References

- Bishop CM (1995) Neural networks and pattern recognition. Oxford University Press, Oxford
 Brause R (1991) Neuronale Netze. Teubner, Stuttgart
 Bronstein IN, Semendjajew KA, Musiol G, Mühlig H (2005) Taschenbuch der Mathematik, 6th edn. Harri Deutsch, Frankfurt a. M
 Kohonen T (1989) Self-organization and associative memory. Springer, Berlin
 Pao YH (1989) Adaptive pattern recognition and neural networks. Addison-Wesley, Reading
 Reilly DL, Cooper LN, Elbaum C (1982) A neural model for category learning. *Biol Cybern* 45:35–41
 Schürmann J (1977) Polynomklassifikatoren für die Zeichenerkennung: Ansatz, Adaption. Anwendungen. R, München
 Smith M (1993) Neural networks for statistical modeling. Van Nostrand Reinhold, New York
 Zell A (1994) Simulation neuronaler Netze. Addison-Wesley, Bonn

Chapter 7

Gauging

Dimensional or shape checking and gauging is one of the most demanding applications of industrial image processing, algorithmically as well as with regard to systems engineering and facility construction. It is actually possible to reach accuracies of just a few light wavelengths, but this requires considerable effort. As in every technical discipline, precise results cannot be achieved without corresponding diligence, especially with regard to peripherals, selection of components, mechanical setup, illumination, and image capture. Quality lost in the sensory chain is lost forever. For this reason, an overview chapter on illumination and image capturing will immediately follow this chapter.

Dimensional checking can take many forms, from relatively coarse measuring of part dimensions to high accuracy gauging in the micrometer range. Measuring can be an inspection task of its own, e.g. checking whether a part stays within metrically defined allowances, but it can also be a tool for assembly checking. In this case, the precise value of a measurement may be less important than relationships between measurements which give information about the correct position and assembly of a component in relation to other components.

Methods and systems developed for optical measuring are correspondingly varied. They range from the determination of relative measurements on the level of image pixels and subpixel-precise gauging as part of standard systems to highly specialized coordinate measuring systems, complete with mechanics, optics and illumination. Such systems can reach accuracies of 1 μm —even more accurate for integral measurements such as an object's centroid—but due to their design they are only applicable for laboratory use or in automated manufacturing lines in a controlled environment (e.g. clean rooms).

Before we start to give a feel for the possibilities and prerequisites of this technology, we need to introduce some general terms and concepts.

7.1 Gauging Tasks

Gauging can serve various different purposes and, correspondingly, makes diverse demands on algorithms and sensors. Typical applications with very different requirements are:

Assembly checking: Here, gauging methods are used to check the presence of components or other features (like bore holes). Gauging thus becomes a tool for presence verification. Demands on accuracy are typically not very high in this area so that simple pixel measuring methods can often be applied.

Shape checking: The purpose of this type of application is to detect deviations from a predefined ideal shape. Some examples are checking whether the edge of a part is sufficiently straight and smooth, whether all pins of an IC are situated on a straight line, whether a group of bore holes is not only present but positioned in an exact circle. This kind of application often requires a high relative accuracy but not a high absolute accuracy. This means that measuring results have to be precisely repeatable and lie within narrow tolerances. A highly accurate calibration for the conversion of pixels to metrical values is less important.

Dimensional checking: The most demanding tasks are in the area of actual dimensional checking where certain measurements have to be determined with high absolute accuracy. This not only puts high demands on the measurement computation itself, which will practically always require subpixel interpolation methods, but also on the sensory chain, which has to guarantee absolute stable and repeatable imaging conditions. Only then a calibration is possible that allows for an accurate conversion of image measurements into real values.

In the following sections we will provide examples of these types of gauging problems. With each example we will introduce the typical methods and algorithms one after the other: pixel measuring, gauging with respect to ideal shapes, and subpixel accurate gauging. Calibration and sensors are discussed in a separate section at the end of the chapter.

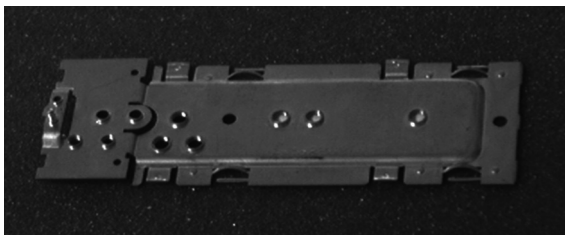
7.2 Simple Gauging

We start our journey into the field of dimensional checking with a very simple example to introduce various aspects of a gauging task without having to go into the particulars of the test piece or the problem itself. In the interest of a concise and self-contained representation, we will accept some simplification. We will come back to this in our discussion of individual measurements and measuring methods.

Figure 7.1 shows the test piece for this introductory example. It is a punched side-panel of a car radio. In the following we will determine some measurements on this part.

Figure 7.2 shows the front portion of the panel in a back-lit image, i.e. in front of an area light. Back lighting is very typical for use in gauging tasks because of

Fig. 7.1 Side-panel of a car radio



the excellent contrast. There are some interesting snags, however, which we will come back to in Sect. 7.7.

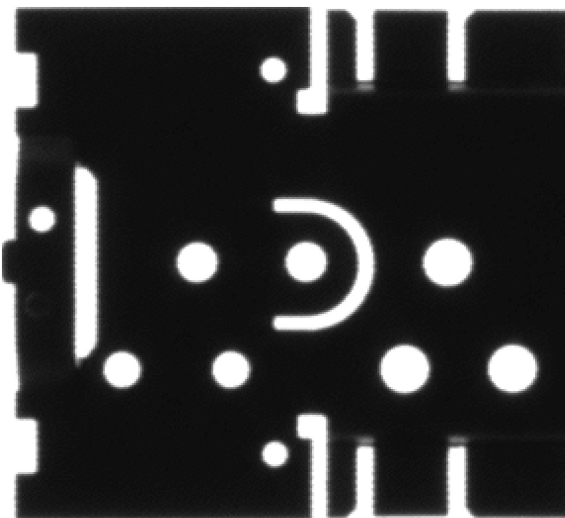
7.2.1 Centroid Distances

There are many ways to measure distances in image processing, with regard to definition as well as methods. Distances can be measured between the centroids of objects, their contours or edges, perpendicular to edges, etc.

One of the simplest tasks is the measuring of centroid distances. As soon as the relevant objects have been found, their centroids can be determined by averaging over all points of each object separately (more about this in Sect. 10.1.3). When the centroids (x_1, y_1) and (x_2, y_2) of two objects are known, the pixel distance can be computed simply using the Pythagorean Theorem:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \quad (7.1)$$

Fig. 7.2 Back-lit section of punched panel



This gives the distance in pixels. If one wants to know something about the real objects, a calibration has to be performed that converts pixels into metrical measurements. Since we discuss this far-from-trivial aspect of gauging in Sect. 7.7, we provide only the essentials here.

Because of perspective distortions, aberrations, etc., scale is not uniform over the entire image area. This fact requires special consideration in high-accuracy gauging. For the moment, we will assume uniform scale over the whole image. We further assume that the pixels are square, i.e. that a horizontal distance of 1 mm is represented by the same number of pixels as an equal distance in the vertical direction. Under these conditions, the metrical distance of the centers of gravity can simply be computed by multiplying the pixel distance from Eq. (7.1) with a calibration factor c :

$$d_m = cd = c\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \quad (7.2)$$

It appears visually obvious that pixels are square, but this is due to the fact that monitors usually display them this way. Actually, non-square pixels were the rule at the beginning of industrial image processing. If we drop this assumption, the conversion becomes a little more complex requiring the application of different calibration factors in x - and y -direction:

$$d_m = \sqrt{(c_x(x_1 - x_2))^2 + (c_y(y_1 - y_2))^2}. \quad (7.3)$$

In all subsequent examples we will not refer to calibration for the sake of clarity. We will cover this topic in Sect. 7.7. Furthermore, we will assume in all subsequent examples that pixels are square. One reason for this is the simplification of the algorithms, which not only concerns the conversion from pixels to metrical measurements but much more importantly the computation of certain measurements themselves, e.g. contour distances which can only be determined with considerable difficulty for non-square pixels. However, the primary reason is that in light of state-of-the-art technology non-square pixels should be regarded as a sensory defect. It is not quite understandable why a sensory defect should be remedied by algorithmic acrobatics. Of all application areas of industrial image processing, gauging certainly puts the highest demands on sensors, from illumination to camera technology to image transmission, and one cannot expect reliable results if the sensory chain is already less than optimal.

Let us return after this excursion into calibration to the centroid distances. What we want to measure is the distance between the two small holes at the top and bottom of the panel. These holes have to fit the corresponding inside threads on the radio casing so that the panel can be fastened to the casing by screws put through these holes. Figure 7.3 shows the measurement to be determined.

Trying out the example described in Table 7.1 you will notice that the distance is computed with decimals. The same is true of the coordinates of the centroids themselves. Computing these also yields decimals. This leads to the question how a measurement on a spatially discrete pixel image can result in a decimal value.

Fig. 7.3 Centroid distance of two holes

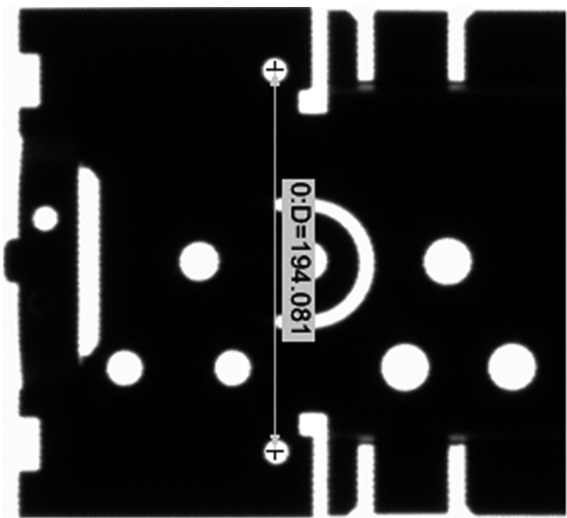


Table 7.1 The following table summarizes the example from Sect. 7.2.1

Category	Description
Index	\\Examples\\Gauging
Illumination	Diffuse back lighting
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define Regions of Interest</i> , full image 3. <i>Determine Threshold</i> 4. <i>Create ROIs by Thresholding</i> , entire panel 5. <i>Create ROIs by Thresholding</i> , all holes within the panel 6. <i>Gauge ROIs</i> , distance of the two holes

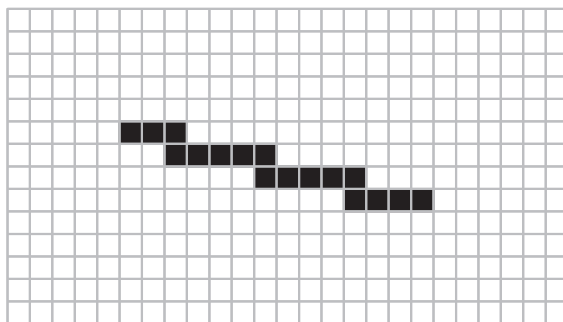
The answer to this question can be found in the way centroids are computed for an object. It is of course correct that the individual pixels of the object, including pixels on the contour, all have whole-number, discrete pixel coordinates. The centroid, however, is computed by averaging over all these coordinates, which makes it more precise than a single pixel. Figure 7.4 shows the simple example of a line running through the image at a 15° angle. The pixel grid shows that the line passes through the following points¹:

(5;5) (6;5) (7;5) (7;6) (8;6) (9;6) (10;6) (11;6)

(11;7)

(12;7) (13;7) (14;7) (15;7) (15;8) (16;8) (17;8) (18;8)

¹ Counted according to programmer custom from (0, 0), top left to bottom right.

Fig. 7.4 Digitized lines

The central point of this spatially discrete digital line is the point (11; 7) printed in bold in the table. Assuming that the end points of the line are exact, the true centroid is at (11.5; 6.5)—and indeed, computation of the centroid by averaging yields (11.4; 6.59), which is only 0.1 pixels away from the true centroid.

Averaging over the entire object has another welcome effect. Centroid measurements are very robust with respect to illumination variations and are therefore ideally suited for calibration. Figure 7.5 shows the effect of varying exposure. The radii of the two holes vary with the exposure between 5.5 and 7.5 pixels, whereas the centroid distances lie within a range of less than 0.2 pixels.

7.2.2 Contour Distances

Often not only distances between objects' centers of gravity are of interest, but also distances between their outer contours. This, however, raises the question of how such a distance can be defined. Mathematically, distances are given between points, and contours consist of many points. Consequently, there is a variety of possible contour distances, depending on how the points used for measuring are selected.

An obvious selection would be the points closest together or farthest apart, giving the minimum or maximum distance between the contours. Interesting applications for the minimum distance between two contours are, for example, the width of the isolation between path conductors on a circuit board or the core diameter of a thread, measured as the minimum distance between the outer contours of the thread, as in the introductory example in Sect. 1.6. The maximum distance of two contours can be used in the same way to determine the outer diameter of a thread. An interesting question with regard to our punched panel could be how close the vertical slit and the circle at its bottom right corner lie together. If this distance becomes too small, the sheet may lack stability. Figure 7.6 shows a magnification of the resulting dimension line.

Computing such a contour distance is much more work than measuring a centroid distance: between every pixel of one contour and every pixel of the other the distance has to be computed, then minimum or maximum value selected. The computation of the measurement depicted in Fig. 7.6 therefore takes almost five



Fig. 7.5 Change of apparent diameter with varying exposure

Fig. 7.6 Minimum distance between two contour points



times as long as that of the centroid distance in Fig. 7.3. Simplifications are very difficult unless one can use prior knowledge about the shape of the objects. If one can assume that the objects are always convex,² then only those parts of the contours on the side facing away from the other object have to be used to determine the maximum distance. If, however, the objects have more complex shapes, possibly enclosing each other, such simplifications are not possible, as Fig. 7.7 illustrates.

The bright lines and crosses in Fig. 7.7 indicate the most distant points on the two contours and the corresponding dimension lines. The points of maximum distance on the contours of the two convex objects on the left are always on the side of the object's centroid facing away from the other object. This is not the case with the dimension line labeled "5" since there the point on the contour of the circle falls in between the centroids of the two objects.

Relay contact clearance: Figure 7.8 shows a close-up of a relay contact. Before installation of the relay it has to be verified that the clearance is neither too large—causing the relay not to pick up at all—nor too small which might result in a flashover. This inspection task is a typical application for the minimum contour distance.

Other distance measurements: Distance measurements can be defined in many ways. Instead of the simple distance, in some cases the distance in a pre-defined direction is of interest. An example is the gauging of the maximum thread

² An object is called convex if all lines connecting any pair of points on its contour run completely inside the object.

Fig. 7.7 Maximum contour distances of convex and enclosing objects

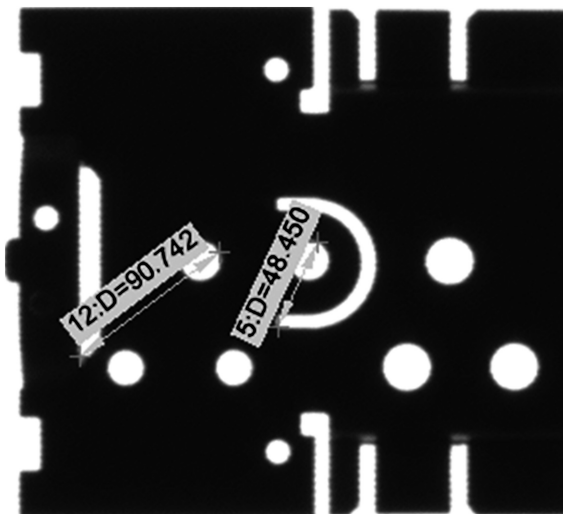


Fig. 7.8 Inspection of relay contact clearances



diameter of the spark plug from the introductory example in Sect. 1.6. Figure 7.9 demonstrates the difference between the simple maximum distance and the directed maximum distance of the two thread contours. The simple maximum distance stretches diagonally from one end of the bottom border to the opposite end of the top border, bearing no relation at all to the desired measurement: the outer diameter of the thread. This can be determined by measuring the distance between the contours in a 90° direction. To determine this measurement, two straight lines at right angles to the desired measuring direction are defined in the image. Each one is shifted until it has exactly one point in common with one of the two contours. The perpendicular distance between the two lines is equivalent to the desired measurement.

Fig. 7.9 Difference between simple and directed maximum contour distances

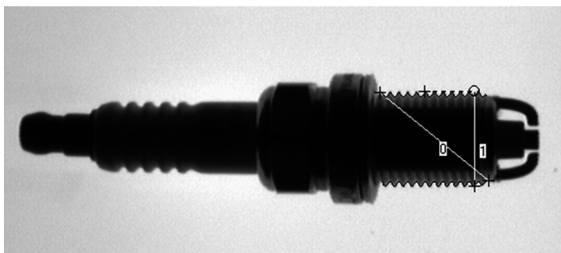
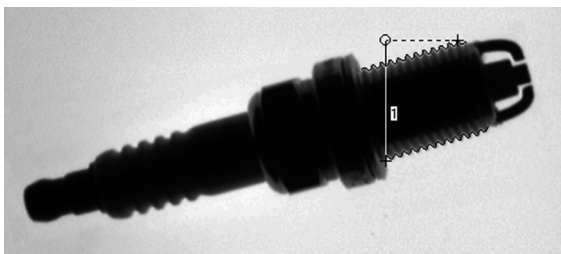


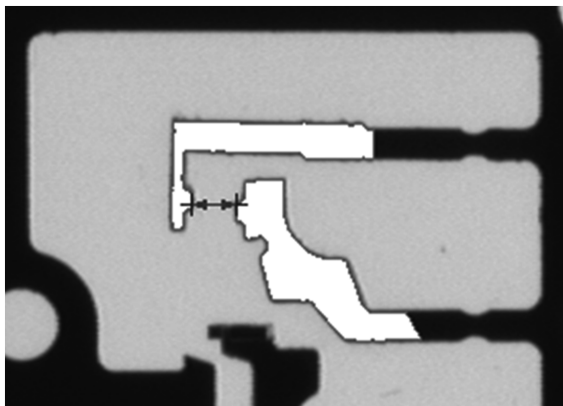
Fig. 7.10 Directed distance measuring without rotation compensation



When computing directed distances it is important to know whether the object has been rotated. Figure 7.10 shows the effect of measuring the maximum distance in 90° direction without taking into account the rotation of the object to be measured.

In some cases, the change of the distance measurement through the rotation of the object may even be a desired effect, as there are many, many different ways of defining distances between contours. We would like to mention one other before turning to entirely different measurements. Figure 7.11 shows a section from a punched part that we will take a closer look at in Sect. 7.3. The thick black line between the contacts emphasized in white indicates the distance to be measured.

Fig. 7.11 Principle and result of a caliper measurement between contours



One could try the method described above to measure a 0° direction, but this leads to a problem. Since the contours overlap, there is not a specific closest or farthest point telling us where to stop shifting the parallel lines. This is a very typical case in the inspection of electronic components, be it switches or circuit boards, where instead of the perpendicular measuring between parallel lines a kind of “edge probing” has to be performed. By this we mean drawing a line from each point of the contour in the defined measuring direction. If this intersects the other contour, a measurement is taken, if not, this line is ignored. This gauging method can be compared to the typical way of measuring inside dimensions mechanically by inserting a sliding caliper into the opening to be measured. Figure 7.11 illustrates both principle and result of this “caliper measurement”.

Program hint: **NeuroCheck** defines the gauging rules “Minimal/maximal distance opposite” for the caliper measurement, the gauging rules “Minimal/maximal distance directed” for the measurement between parallel lines perpendicular to the measuring direction.

7.2.3 Angle Measurements

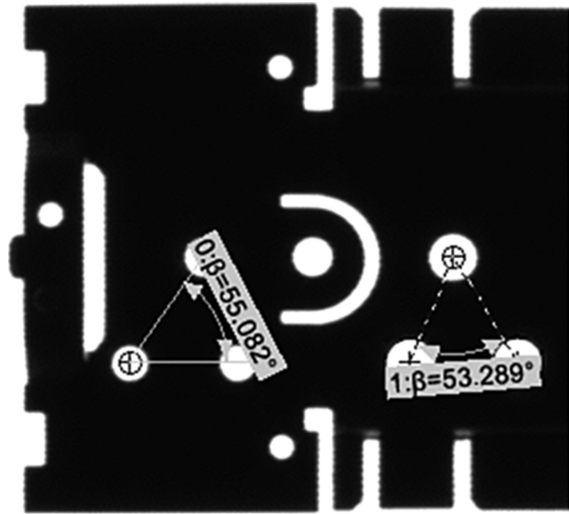
Angles can, for example, be used to check the position of objects in relation to each other. The simplest method of measuring an angle in an image is based on defining a straight line by two points in the image. The direction of this line can be described by its angle towards the horizontal axis. As soon as the centroids of two objects are known, a straight line can be drawn through these points and its angle can be computed according to Eq. (7.4) where Δx and Δy are the differences of the x- and y-coordinates of the two centroids. Since the tangent function is π -periodic, some cases have to be distinguished. Also one has to intercept the special case $\Delta x = 0$, in which the angle is $\pi/2$ for $\Delta y \geq 0$, else $-\pi/2$, but apart from these special cases it is a straightforward measurement.

$$\alpha = \arctan\left(\frac{\Delta y}{\Delta x}\right). \quad (7.4)$$

Figure 7.12 shows two examples for this method. In the first example the centroids of the two punched holes on the left were connected by a line whose direction was then computed as the angle towards the horizontal. In the second example the centroids of the two bottom holes of the triple on the right were connected with that of the top hole. Then the aperture angle of the two rays was computed.

A more difficult type of angle measurement deals with gauging the direction of an object contour. We will discuss this in the following two sections.

Fig. 7.12 Angle measurements on punched side-panel



7.3 Shape Checking on a Punched Part

7.3.1 Inspection Task

Figure 7.13 shows a section from a punched part. One of the inspection tasks on this part was to detect bent contacts which would lead to a malfunction of the finished product.

The parts are transported across a back light; since they are guided on both sides and transported in steps, it is very simple to capture a precisely positioned still image from each element: the image is taken the moment the parts are stopped for an operation. The signal indicating that a new element has been brought into position under the camera is simply sent to the image processing program, which then freezes the current camera image and processes it. Position adjustment is not required, the search areas can be defined fixed in the positions shown in Fig. 7.14.

Along these search lines, edges are detected by searching for the first light-to-dark transition and tracing the contour to the limits given by the neighborhoods indicated in the left image of Fig. 7.14. The detected edges are emphasized white in the right image of Fig. 7.14.

A closer look at the defective element in Fig. 7.13 reveals the actual inspection task: the decisive criterion is the angle of the contact to the horizontal. This angle measurement differs from those in Sect. 7.2.3 in an important aspect: there the directions were always given by straight lines defined by two points; here it is the angle of a contour which actually changes its direction at several points. The following section deals with the question of how to determine the angle of a contour.

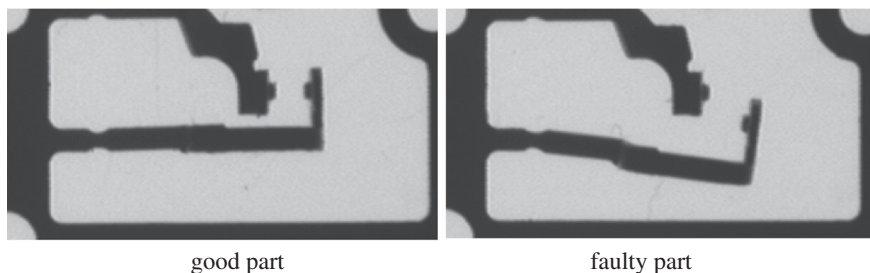


Fig. 7.13 Back-lit punched part

7.3.2 Modeling Contours by Lines

The angle measurements in Sect. 7.2.3 were very simple because we always had two centroids directly defining a straight line, i.e. something with a clearly defined mathematical description. In this application, in contrast, we deal with a contour which in the end is nothing but a collection of individual pixels. The tangent to the detected contour in Fig. 7.14 has different angles at different points of the contour since it changes its direction several times.

This raises the question of how to define the direction of a contour or an edge at all. The answer is intuitively clear. A human would—figuratively speaking—align a ruler as closely as possible to the contour and measure its angle. In other words, he would try to approximate the irregular edge by a straight line. Exactly the same thing can be done digitally in the image. We will refer to this line by the term *model line* since we use it to model the actual contour. It is equivalent to what statisticians (Bronstein et al. 2005) call a regression line, which is defined as that line for which the sum of the squared vertical distances of all data points is minimal. We will introduce the method as it is defined in statistics, since this makes equations a little more simple. Then we will present some variations necessary for practical image processing applications.

Figure 7.15 illustrates the definition of a regression line using a set of n data points to be approximated by a straight line. The straight line is generally described by

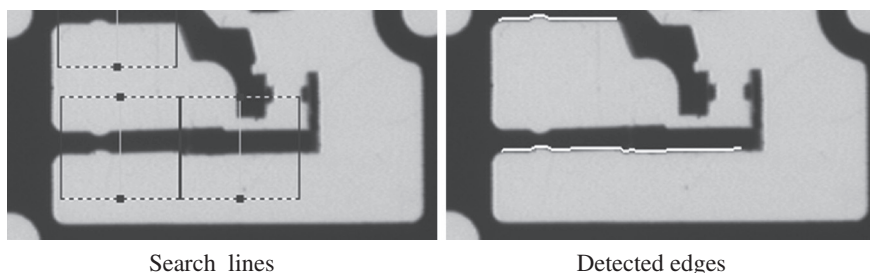
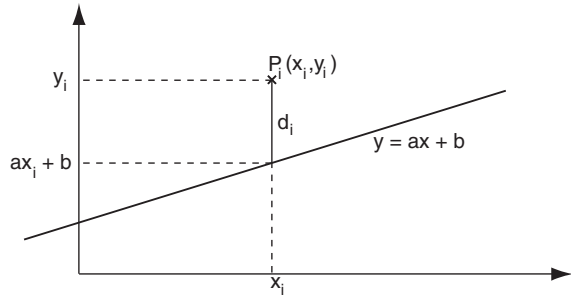


Fig. 7.14 Search lines for punched part inspection and detected edges of the contacts

Fig. 7.15 Computation of a regression line



$$y = ax + b. \quad (7.5)$$

For each point $P_i = (x_i, y_i)$ the vertical distance from this line is

$$d = y_i - ax_i - b. \quad (7.6)$$

The sum of the vertical squared distances of all n points from the line is therefore

$$D = \sum_{i=1}^n (y_i - ax_i - b)^2. \quad (7.7)$$

The coefficients a and b are to be determined such that this sum is minimal. The necessary condition for a minimum is that the partial derivatives of the sum with respect to both coefficients are zero. The partial derivatives can be taken under the sum because of the linearity of summation, leading to the following conditions for a minimum:

$$\frac{\partial D}{\partial a} = -2 \sum (y_i - ax_i - b) x_i \stackrel{!}{=} 0, \quad (7.8)$$

$$\frac{\partial D}{\partial b} = -2 \sum (y_i - ax_i - b) \stackrel{!}{=} 0, \quad (7.9)$$

From these conditions the well-known equations for linear regression are derived:

$$a = \bar{y} - b\bar{x}, \quad (7.10)$$

$$b = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2}. \quad (7.11)$$

where $\bar{x}$ and $\bar{y}$ are the mean values of the x resp. y coordinates of the data points.

We mentioned above that one has to proceed a little differently for practical image processing purposes. The fundamental method does not change; the sum of the squared distances is still minimized, but the distances themselves have to be computed differently for the following reasons:

1. Vertical regression lines cannot be computed with the above method.
2. Better results for image processing applications are obtained by using the perpendicular distances between data points and line instead of the vertical distance.

Iterative computation: Usually, one begins by computing a regression line as described above. Then the coordinate system is transformed in such a way that this initial regression line becomes the x-axis. In this coordinate system the vertical distances of the individual points are practically perpendicular to the axis. By performing another regression and transforming the coordinate system back to its original orientation, the approximation can be improved. The problem of vertical regression lines is also solved by this method since after the coordinate transformation an approximately horizontal regression line is computed. The final vertical line is created by the inverse transform.

The iterative computation scheme also shows a way to suppress the influence of “outliers”. A small irregularity in the object contour can have a strong effect on the regression line depending on where the irregularity occurs. Such disturbances can, for example, be created by surface texturing. To suppress their influence one can remove all points in each iteration that are too far from the current regression line. The smaller a permissible distance of the points from the current regression line is chosen, the faster the iteration will converge to a specific direction. This will typically yield a very good approximation of the perceived axis of an object. If the permissible distance is set too small, it can happen that a poor initial approximation causes points to be thrown away which are actually relevant but too far from this poor initial line.

7.3.3 Measuring the Contour Angle

After making this excursion into the computation of model lines let's return to the inspection of the contact contours. After computing a separate model line for each contour section using the method described above, the slope of these model lines is equivalent to the tangent of the desired angle. Comparing this to the tolerances given for the part solves the inspection task. Figure 7.16 shows the dimension lines for a good switch and a defective switch. Table 7.2 summarizes the example.

7.4 Angle Gauging on Toothed Belt

Figure 7.17 shows a common toothed belt. The inspection task was to determine the angle between the teeth and the edge of the belt before the belt is cut and formed in a ring. Deviations between the nominal angle—given by the gear—and the actual angle of the teeth on the belt would lead to unquiet running and excessive wear. An accuracy of 0.5° was required in this task.

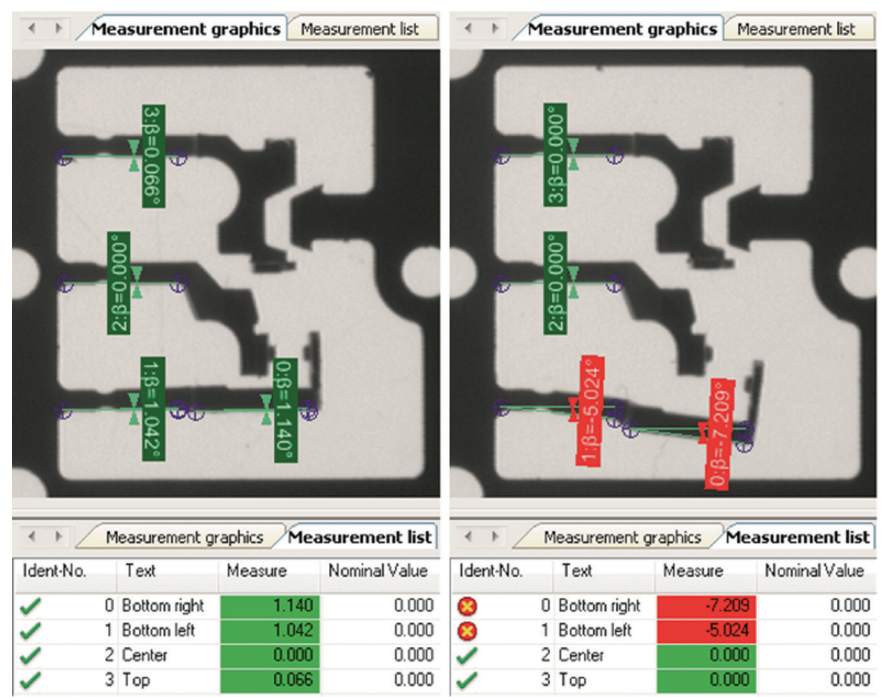


Fig. 7.16 Measuring results on punched parts (measurements within tolerances *green*, outside *red*)

Table 7.2 The following table summarizes the example from Sect. 7.3

Category	Description
Index	\\Examples\\Gauging\\
Illumination	Diffuse transmitted lighting with back light
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, search lines for probing the edges of the individual contacts 3. <i>Determine Threshold</i>, optimizes the threshold using a histogram analysis 4. <i>Create ROIs by Thresholding</i>, detects and traces the contours of the contacts 5. <i>Compute Model Geometries</i>, computes a regression line for each traced contour section 6. <i>Gauge ROIs</i>, computes from the parameters of the regression lines the angles of the individual contour sections 7. <i>Check Allowances</i>, compares the computed angles with the nominal values

7.4.1 Illumination Setup

The main problem in this application is to obtain a clearly defined edge for measuring the angle. The left image in Fig. 7.18 shows that a normal front lighting

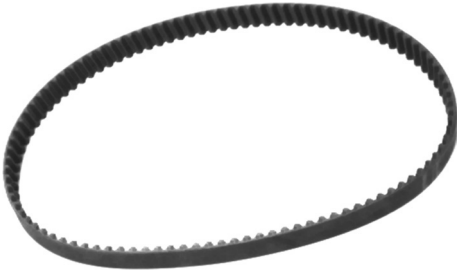


Fig. 7.17 Toothed belt (heavily overexposed to show the teeth on the inside)

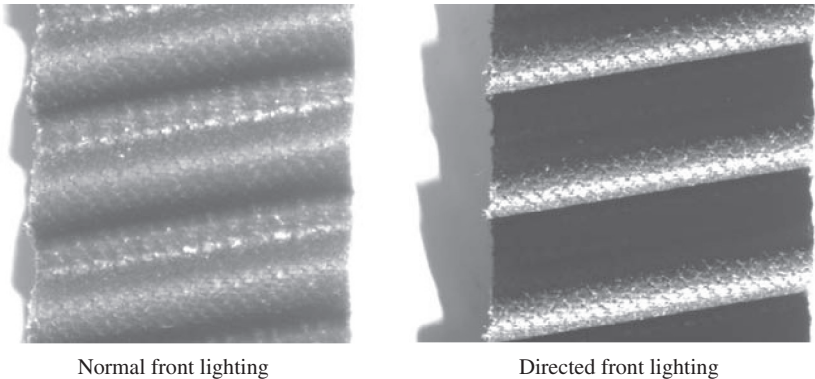


Fig. 7.18 Toothed belt under normal front lighting and under directed front lighting

will not be successful here. The heavily textured material prevents a clear edge segmentation. Instead, the belt has to be viewed under directed front lighting at a shallow angle, causing every tooth to project its shadow onto the next tooth. The result can be seen in Fig. 7.18.

Figure 7.19 illustrates the principle of this shallow directed lighting, viewed from the side. Clearly the level where the shadow will be projected depends on the following dimensions:

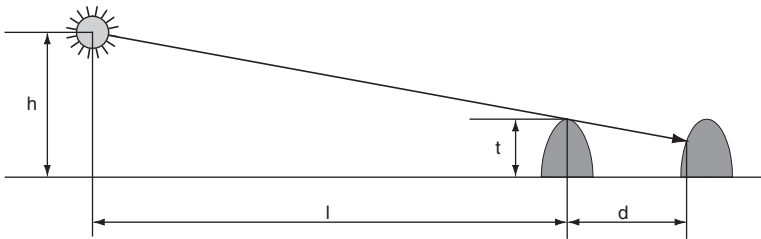


Fig. 7.19 Illumination setup for toothed belt

- Height h of the light source above the level of the toothed belt
- Distance l between light source and lighted tooth
- Height t of a tooth
- Distance d between teeth.

If the rays run parallel to the edges of the toothed belt, the distance between light source and lighted tooth will be different between the left and right edge of the belt because of the angle between belt edge and teeth. This would cause the projected height of the shadow to be different; the shadows would not run parallel to the upper edges of the teeth. The lighting therefore has to be mounted such that the angle of incidence of the light rays is perpendicular to the orientation of the teeth. For a correctly oriented tooth, the projected shadow will then be parallel to the ridge of the next tooth. If one of the teeth deviates from the nominal orientation, its projection will show a corresponding deviation.

Note that there can be two failure modes superimposed: the shadow is created on a possible faulty tooth and is projected onto another tooth which can also be faulty. In this application the accuracy achieved by the above setup met the specifications. For higher accuracy, the first cause of errors can be removed by generating the projected shadow with a defined obstruction in the ray path, for example a precisely manufactured and aligned metal edge. In this setup, the shadow line will only reflect the deviations of the tooth onto which it is projected.

7.4.2 Edge Creation

In order to gauge the angle we need a straight line. In principle, this could be generated in the same way as in Sect. 7.3, i.e. by computing a regression line from all points of a thresholded edge contour. But even under optimized illumination, the material of the toothed belt still shows heavy texturing that makes edge segmentation via thresholding extremely difficult. The edge detection method described in Sect. 4.4 is far superior in this case. It regards a pixel as part of an edge if a specified gray level gradient is detected in a neighborhood of that point. This neighborhood can then be examined further using the interpolation methods also discussed in Sect. 4.4 to determine the location of the edge point with sub-pixel accuracy. Thanks to the high level of accuracy possible with this method, it is sufficient to detect edge points along a limited number of search lines instead of examining the entire edge, thereby greatly reducing the computation effort.

Maximum stability for the edge detection is achieved in this example by searching for the first rising edge out of the shadow area. This allows for precise location of the projection line on the next tooth. On each of the search lines drawn brightly in the left image of Fig. 7.20 exactly one subpixel-precise edge point is detected. From all these points, a regression line can be computed in exactly the same way as in Sect. 7.3.2. The angle of this regression line can be easily determined. The principle is shown in the right image of Fig. 7.20.

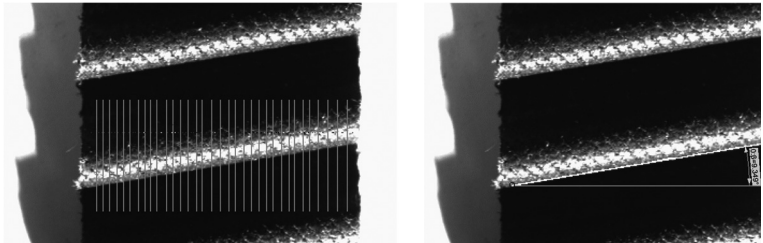


Fig. 7.20 Search lines for edge detection on the toothed belt and the resulting angle measurement

Table 7.3 The following table summarizes the example from Sect. 7.4 without position adjustment

Category	Description
Index	\\Examples\\Gauging\\
Illumination	Low angle of incidence front light adjusted to skewing of teeth of the toothed belt
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i> , sets search lines extending from the shadowed area over the next projected edge. If triggering is not sufficiently precise, search lines have to be readjusted relative to the position of an edge found by thresholding 3. <i>Create Edges</i> , determines the transitions between the shadowed area and the lighted area of the next tooth with subpixel accuracy 4. <i>Compute Model Geometries</i> , computes a regression line from the detected edge points. A small number of erroneously detected edge points can be eliminated by iteration 5. <i>Gauge ROIs</i> , computes the angle of the regression line

Table 7.3 summarizes the application in the form described above. It should be mentioned that the application is sensitive to positioning in this form. If the image is not captured at precisely the correct moment, the search lines move relative to the image so that edges within the heavily textured area of a tooth may be detected. Figure 7.21 shows such a shift of the search lines. In this case, a position

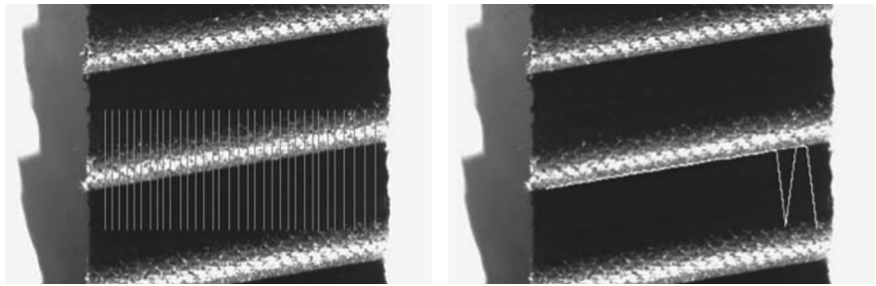
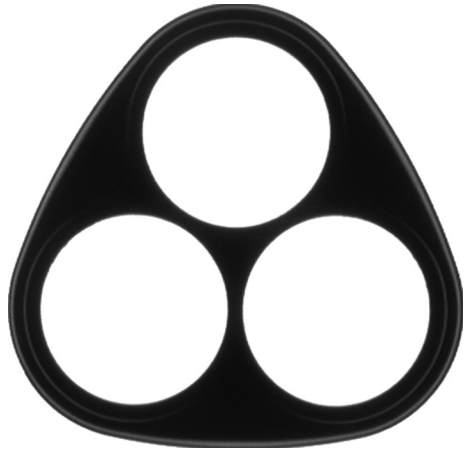


Fig. 7.21 Erroneous edge detection due to position shifts

Fig. 7.22 Back-lit injection-molded part



adjustment is required. A thresholded edge is sufficiently stable for the positioning process. We will not go into that here because the position adjustment can be performed in the same way as, for example, in the applications in Chap. 5.

7.5 Shape Checking on Injection-Molded Part

Object contours can be modeled not only by straight lines, as in Sect. 7.3, but also by other ideal shapes, for example circles. This allows for measuring radii and deformation. Figure 7.22 shows a plastic part in which the radii of the rounded corners and the arrangement of the holes has to be inspected to detect a possible deformation.

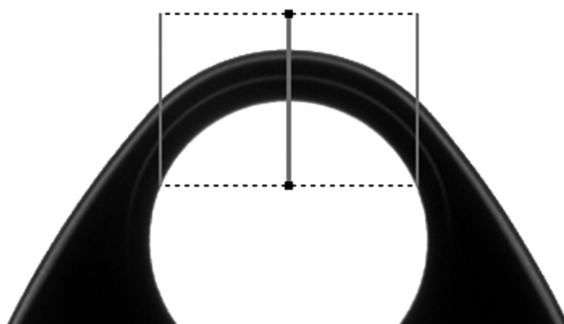
7.5.1 Computing Radii

In order to determine whether the test piece is warped as a whole, the relationships between the three holes can be measured. These are very easy to find by thresholding in the back-lit image. After computing their centroids, their relative positions can be described with the following measurements:

- distances of centroids, as in Sect. 7.2.1
- angle between connecting lines, according to Sect. 7.2.3
- radius of a circle through the centroids.

The first two measurements contain nothing new with respect to Sect. 7.2; the third measurement is also easily determined since a circle is defined precisely by three points. More interesting is the measuring of the radii at the rounded corners.

Fig. 7.23 One of the search lines used for segmenting the rounded portions of the contour

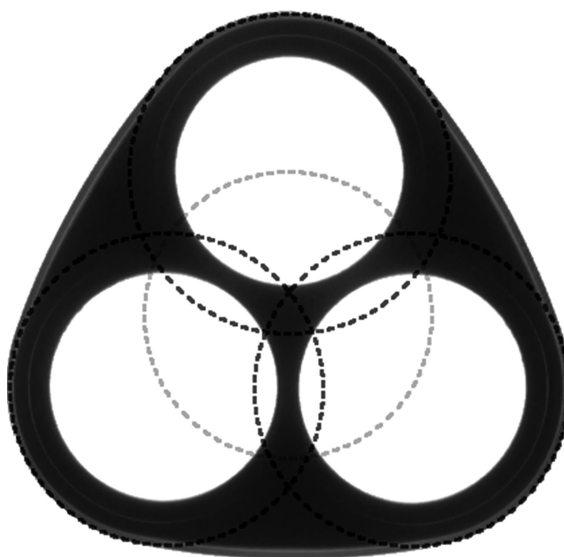


Since these are only circle segments, entire circles must first be aligned to these contours.

To this end, parts of the rounded contours are segmented. One of the search lines used for this purpose is shown in Fig. 7.23. The search lines should pass through the apex of the rounded contour parts, cutting out contour portions which are not too wide in order to avoid including parts of the almost straight side portions in the computation of the circle. Therefore, a position adjustment should precede segmentation. Similar to Sect. 3.3.4, the most stable position reference in this example is a group of objects, here formed by the three holes.

Complete circles can then be fitted onto the segmented contour portions. Apart from the fact that three parameters (x - and y -coordinate of the center and the radius) have to be optimized, this computation is performed according to the same principle as for the regression line in Sect. 7.3.2, so we can skip the theory here. Figure 7.24 shows the resulting circles for the three corners and the circle through centroids of the holes

Fig. 7.24 Model circles for rounded corners and circle through centroids of the holes (circle lines thickened)



the centroids of the three holes. It can be seen that the model circles conform precisely to the shape of the rounded corners, moving away from the object contour at the point where the curves become almost straight side portions.

As soon as the parameters of the circle have been determined, the radius of the corner is also known since it is one of those parameters. In addition, the minimum and maximum distances of the actual contour from the computed circle center can be determined. The difference of these two values is also denoted as *roundness* and can be used as a measure for the similarity of a contour to a circle. An ideal circle has a roundness of zero. This measurement can be used here to evaluate deformations of the outer contour. Deformations of the entire test piece will be noticeable from the radius of the circle through the centroids of the three holes. The roundness of this circle, incidentally, will always be precisely zero since three points define an exact circle. No regression is necessary in this case. The inspection procedure is summarized in Table 7.4.

Table 7.4 The following table summarizes the example from Sect. 7.5

Category	Description
Index	\\Examples\\Gauging\\
Illumination	Diffuse back lighting
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, simple search area to detect the entire part 3. <i>Determine Threshold</i>, can simply be set manually because the back lighting creates very stable brightness conditions 4. <i>Create ROIs by Thresholding</i>, segments the entire part as an object 5. <i>Create ROIs by Thresholding</i>, segments the three holes within the part 6. <i>Determine Position</i>, computes the current position and orientation of the part from the group of holes 7. <i>Define ROIs</i>, puts search lines on all three corners of the reference part 8. <i>Position ROIs</i>, positions these search lines according to the computed current position 9. <i>Create ROIs by Thresholding</i>, segments a section from the contour of the part at each of the rounded corners 10. <i>Combine ROIs</i>, combines the segmented holes and the contour sections in a single list of regions for joint gauging 11. <i>Compute Model Geometries</i>, computes the exact circle through the centroids of the three holes and the best-fit circles for the contour sections at the three rounded corners 12. <i>Gauge ROIs</i>, computes the radii of all these circles, also the minimum and maximum distances of the contour sections from the centroids of the respective best-fit circles to detect deformations

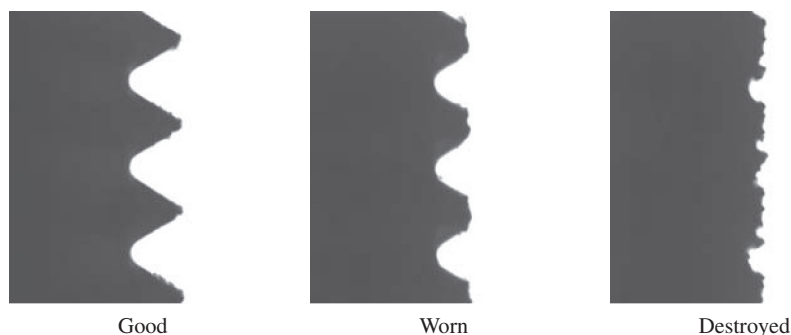


Fig. 7.25 Thread flanges cut with tools in different states of wear

7.5.2 Comments on Model Circle Computation

As with model lines, an iterative procedure can be used for computing model circles. In each iteration, those points will be discarded which exhibit the greatest distance from the circle. In our example here, this can be used to reduce the influence of contour points from the nearly straight side portions of the test piece.

Model circles can be computed from individual points instead of a contour, just as it has been done for a model line in Sect. 7.4. Iterative computation of a medium circle from a set of subpixel-accurate contour points is a very accurate method to determine the diameter of a circular work piece or a bore hole.

7.6 High Accuracy Gauging on Thread Flange

This application is concerned with a thread flange manufactured through complete automation on an NC lathe. Cleaning and checking of the product is also fully automated. The inspection task was to measure the thread depth with a high level of accuracy to enable early detection of cutting tool wear. If wear is not detected in time, unusable threads may be produced. In extreme cases the tool can break and cause further damage. Figure 7.25 shows the cutting result for various states of the tool. The originally sharply defined thread shape appears worn down due to increasing tool wear until, at the end, only a destroyed blank is left.

The required accuracy of $2\text{ }\mu\text{m}$ is close to the light wavelength. Taking into account that the nominal thread depth is 1 mm and that some play is necessary to compensate for position variations and to probe the edge of the thread contour, one can easily see where the challenge lies in this application. First, the image quality required for this level of accuracy puts the highest demands on illumination and sensor equipment. On the other hand, using a camera with 2448×2050 pixels requires an algorithmic accuracy of approx. $1/3$ of a pixel since a field of view of approx. 10 mm is needed.

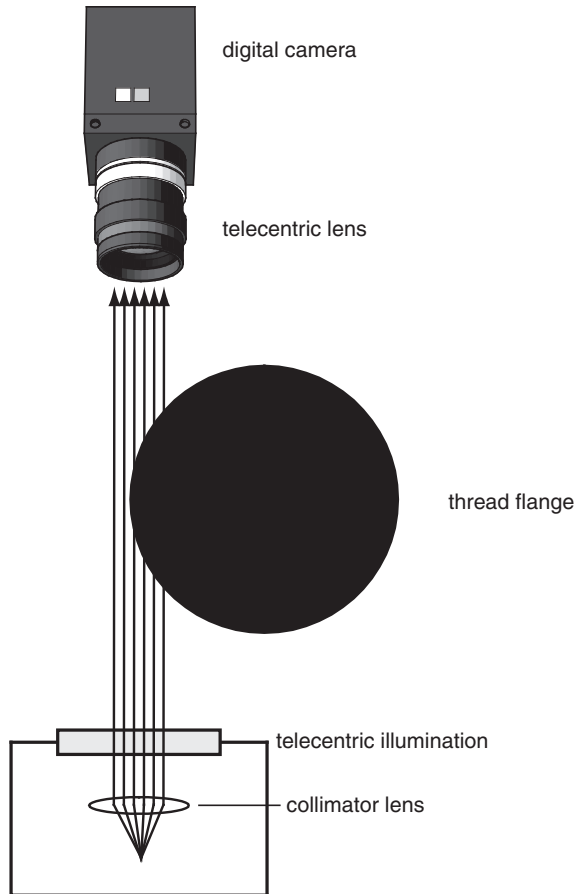
7.6.1 Illumination and Image Capture

Here, as in most gauging applications, illumination and imaging technology are the decisive components. There is surprisingly little freedom of design since the requirements and the physical constraints practically prescribe what type of equipment has to be used:

Illumination: The required sharpness of contours can only be achieved by back lighting. Frequently, one will choose LED illumination with stabilized power supply to avoid variations in brightness. Otherwise, beating waves will be created between illumination and camera rendering stable and reproducible subpixel computation impossible.

Another problem arises in distortions of the contour through lateral illumination common with standard back lighting. This effect can even be seen on the punched part from Sect. 7.2 if it is slightly overexposed, although it is placed flat on the back light. Figure 7.26 shows that in this application the lighting plane will

Fig. 7.26 Schematic top view of the measuring setup



necessarily be at a distance from the measuring plane at least equal to the radius of the flange. The effect of lateral illumination would be much stronger in this case, suggesting telecentric back lighting with a parallel ray path.

Lens: The lens has to be precisely focused on the measuring plane in this application. Small variations of distance must not result in a change of scale. Therefore, only a telecentric lens can be used here.

Camera: The closer the resolution of the image is to the required accuracy, the more stable the measurement will be, because more “genuine” information is available—in contrast to reconstructed information. The camera should therefore have as high a resolution as possible.

7.6.2 Subpixel-Accurate Gauging of the Thread Depth

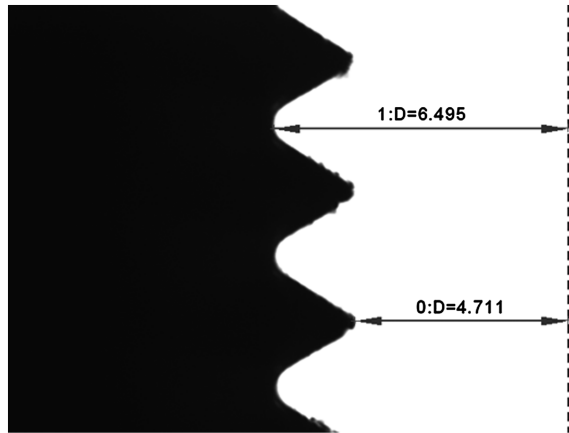
The description of illumination and image capturing situation in the previous section makes it clear that a measuring task with the accuracy required in this application demands a correspondingly precise mechanical setup. Subsequent position adjustment in the image processing system will hardly be possible since a misalignment of the test piece relative to illumination and camera will already result in distortions due to the altered ray path and focus plane. We will have to assume therefore that the test pieces are positioned with absolute precision within the camera’s field of view. This concerns the vertical positioning in particular.

Under this assumption, we can proceed as follows: an artificial, exactly vertical reference line is created; from this line, the minimum and maximum horizontal distance to the thread contour is measured. The highest theoretically possible accuracy can be achieved by performing a subpixel interpolation of the entire contour, but the telecentric ray path of illumination and lens, the stabilized lighting and digital image transfer create sufficiently stable imaging conditions to restrict the subpixeling to those points of the contour closest to and farthest from the reference line. In a neighborhood of these points, the subpixel interpolation is performed. Then the exact distance from the line can be determined. Figure 7.27 shows the measuring lines for determining the thread depth. The application is summarized in Table 7.5.

7.7 Calibration

We have mentioned briefly the important topic of calibration in Sect. 7.2, but have neglected it in the subsequent examples to achieve a more concise and straightforward presentation of the aspects of optical measuring. In these examples, we assumed that:

Fig. 7.27 Measuring lines for thread depth



1. the sensory equipment (optics, camera) guaranteed a constant, uniform imaging scale;
2. the image was created with the correct aspect ratio, i.e. the camera had square pixels and the transfer to the computer maintained this aspect ratio, be it by pixel-synchronized sampling or by digital transmission;
3. the conversion factor between the size of a pixel and the dimensions in the real world had been determined from a suitable reference part.

Chapter 8 discusses the first two aspects in detail; the third one will concern us in this section. The key phrase hereby is “suitable reference part” since after measuring a dimension which is precisely known in the real world with the highest possible optical precision, the computation of the conversion factor is mathematically trivial. The question is therefore, how to obtain a suitable measurement, or, in other words, what can be used as reference part and how its image should be captured and gauged.

7.7.1 Calibration Mode

Many image processing systems have a special calibration mode in which a calibration target, i.e. a reference part with certain predefined characteristics, is used to determine the imaging properties of the system. Such a calibration target can be:

- Glass plates with etched markings
- Metal parts, milled and ground to precisely specified dimensions
- Metal plates with precisely drilled holes.

Table 7.5 The following table summarizes the example from Sect. 7.6

Category	Description
Index	\\Examples\\Gauging\\
Illumination	Telecentric back lighting
Image capture	Telecentric lens, exactly focused on the midplane of the flange; high-resolution digital camera
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, sets the search line for the thread contour and the artificial vertical reference line 3. <i>Determine Threshold</i>, optimizes the threshold by histogram analysis 4. <i>Create ROIs by Thresholding</i>, performs initial segmentation of the contour line with pixel accuracy as starting point for the subpixel algorithm 5. <i>Combine ROIs</i>, combines the vertical reference line and the thread contour for joint gauging 6. <i>Gauge ROIs</i>, computes the minimum and maximum horizontal distance between the reference line and the thread contour. First it searches for the points on the contour having minimum and maximum pixel distance from the reference line. Then it performs the subpixel interpolation in a neighborhood of these points 7. <i>Derive Measurements</i>, computes the thread depth as the difference of maximum and minimum distance

The mechanical precision of these calibration targets is undisputed. Nevertheless, we see some problems in this approach, the most significant being that a calibration should be performed with the identical setup as the final inspection. The image capturing situation should not differ between calibration and application. Different illumination or aperture settings will inevitably lead to changes in the image formation process and hence in the calibration. If the size of the calibration target differs from that of the test piece, so that a different field of view, different illumination or different lens is required, the results will not be applicable to the application situation. This in turn may require a special calibration target for every application.

Even if images of the calibration target can be taken at the same distance as the test part using the same lens and illumination, it can differ from the test piece in such a way that the image properties are changed. An important aspect, for example, is the question whether outer edges are to be measured, e.g. the width of a part, or inner edges, like the diameter of the bore hole. Depending on the illumination and the overall setup, image properties important for the calibration may differ between these situations. Or it may not be possible to determine the known reference dimensions of the reference part using the same segmentation methods or the same parameters used later in the actual inspection. The focal plane where the essential properties are clearly recognizable may differ from that of the test piece, again changing image properties...

7.7.2 Inspection-Related Calibration

The considerations from the previous section suggest that calibration should not only take place in the same image capturing situation as the final inspection, but also using the actual test piece. This requires one, preferably several, “master pieces”, test pieces on which the properties to be measured have been determined as accurately as possible using a different method—for instance, a mechanical method which may be very precise but unsuitable for automated production purposes. These master pieces can then be used to calibrate the image processing system *for precisely the task at hand*. The calibration has to be performed using the identical system setup, segmentation and measuring methods and parameter settings as the final inspection. Only this way the applicability of the determined conversion factor for the inspection task can be ascertained.

If it is actually impossible to take the reference dimensions directly from the test piece, the second best way is to manufacture a special calibration work piece whose properties are as similar to those of the test piece as possible. If the objective is gauging the outer edges, a high-precision part of basically similar but simplified shape will be suitable; if the test piece is flat and straight, one would use a flat calibration piece with straight edges, if it is round, one would use a cylinder since contour images of round parts and sharp-edged objects appear quite different. The thread flange application in Sect. 7.6 is a case in point: it required very precise focusing on the midplane of the circular part and use of a rather small depth of field.

An excellent reference piece for the gauging of distances is a metal plate with several bore holes. With modern CNC machines such a reference piece can be manufactured to very high precision with a geometry and size adapted to the test piece. In back lighting, the distance of the centroids of the holes is largely independent of exposure and segmentation since a widening or narrowing of the bore holes due to varying illumination etc. will be symmetrical and will therefore not affect the centroid coordinates.

7.8 Summary

Gauging tasks: Three types of gauging tasks can be distinguished:

- Assembly checking with relatively low accuracy requirements
- Shape verification, focusing on relations between objects and their relative dimensions
- Dimensional checking requiring highest accuracy and correspondingly precise sensory equipment.

Procedure: Gauging applications are in most cases based on edges. Therefore, they follow a typical sequence:

1. Definition of search lines, if necessary after prior positioning
2. Probing of edges, either directly using an edge model or using an initial thresholding stage followed by subpixel interpolation
3. Computation of model geometries to determine angles and radii
4. Computation of measurements
5. Inspection-related calibration
6. Checking of allowances.

Centroid distances: These belong to the most robust and most easily computed measurements; they are naturally subpixel accurate since the centroid is determined by averaging over the entire object.

Contour distances: Distances between contours require a specification between which points and, if necessary, in which direction the measurement has to be performed. For example, minimum and maximum distances can be computed in a pre-defined direction for overlapping contour portions only.

Angle gauging at object centroids: This is the simplest type of angle measurement; it connects centroids of objects by a straight line, the angle of which directly follows from the equation of the line.

Angle gauging on contours: Measuring contour angles requires approximation of the contour by a regression line; iterative computation can improve the approximation through successive removal of outliers.

Model geometry: Contours can be approximated by various geometries, like straight lines, circles and ellipses to determine the deviation from these ideal shapes. Fitting can also be based on individual points of a contour.

Edge detection: Gauging tasks are frequently edge-based; therefore, direct evaluation of edge properties is often preferable to binary thresholding with subsequent subpixel interpolation since the binary threshold may change the starting point of the subpixel interpolation.

Subpixel accuracy: Measuring accuracies better than a single pixel can be achieved either by averaging over a large set of points, as with the centroid computation, or by direct interpolation of gray level profiles.

Illumination: High-accuracy gauging requires stable illumination. Back lighting creates good contrast and sharply defined edges; perspective distortions can be avoided by using telecentric illumination.

Image capture: For gauging applications, square pixel cameras and digital cameras should be used to avoid sampling errors. For high accuracy, telecentric lenses should be used.

Calibration: Determination of the relationship between dimensions in the image and in the real world should be performed under the same conditions prevalent in the final application, if possible using the actual test piece to capture its influence on the image properties. If this is not possible, reference pieces with characteristics similar to the test piece should be preferred to general-purpose calibration targets.

Reference

Bronstein IN, Semendjajew KA, Musiol G, Mühlig H (2005) Taschenbuch der Mathematik, 6th edn. Harri Deutsch, Frankfurt am Main

Chapter 8

Overview: Image Acquisition and Illumination

In the previous chapters we have encountered a number of ways to process digital images, to recognize objects therein and to evaluate them. So far, however, we have not explained how the digital image is acquired whose properties and quality are of the utmost importance in solving an image processing task.

Digital images can be created by completely different sensors. The most frequent case is an image of visible light passing through a lens onto a camera sensor whose signal is then transferred to the image processing system.

All parts of this sensor chain—starting with the lens and the light-sensitive elements and leading to the transmission to the computer—affect the quality of the digital image. Hence, they are decisive factors for the quality of the results that can be obtained from the image. Quality loss incurred here can only be compensated for in a small way by software technology or not at all. All application areas of digital image processing depend on high quality images, but none more so than optical gauging, because of the high demands on accuracy in this area.

In the application examples of the previous chapters we have already seen that illumination is at least as important as lenses and acquisition technology for successfully solving image processing tasks. It is naturally very difficult to give recommendations for illumination setups. Too much depends on the interaction between lighting, test piece appearance, and conditions at the place of installation: space, movement of parts, ambient light etc. Successful experimenting, however, requires fundamental knowledge of lighting techniques. Section 8.6 gives an overview of typical lighting methods and their applications in order to provide a feel for how to approach a given illumination problem.

8.1 Solid-State Sensors

We will begin this chapter on image acquisition technology with the light-sensitive elements themselves; i.e. we assume for the moment that the optical elements have already taken care of forming a high-quality image on the sensor. Thus, we can refer to the properties of the sensors in our discussion of the fundamental laws of

optics in Sect. 8.5 and present the interdependencies of optical and sensor elements more easily.

The first section will give a brief overview of the various sensor types; the properties common to all sensors will be discussed in the first subsection. Building on that, the subsequent sections will discuss CCD and CMOS sensors in more detail. Special sensors and technology will be briefly introduced in the fourth section. Whilst the previous sections dealt with monochrome sensors, a separate section will discuss color sensors and their properties. In conclusion, the general electro-optical properties of all sensors will be presented. For a detailed discussion of the physical aspects, please refer to e.g. (Theuwissen 1996).

8.1.1 Introduction

Solid-state sensors are the electronic eyes of today's cameras. The intensity of the incident light is converted into a charge or a digital value that is subsequently processed depending on the application. Basically, image sensors are categorized according to:

- The geometrical arrangement of the light-sensitive elements (pixels) in line or matrix sensors.
- The mode of operations and manufacturing technology in Charge Coupled Device (CCD) and Complementary Metal Oxide Semiconductor (CMOS) sensors.

Matrix sensors: Matrix sensors like common digital cameras for the consumer market, acquire an image consisting of several lines and columns during a single exposure time. This is the most commonly used type in machine vision by far. Figure 8.1 shows a camera with matrix sensor on the left.

Line sensors: Line sensors acquire only one line at a time. In many line-scan camera applications, lines captured in sequence are assembled into a rectangular image. Figure 8.1 shows a camera with a line sensor on the right. We will discuss the specifics of using line-scan cameras in Sect. 8.4.

CCD sensors: CCD sensors still dominate industrial image processing today; the first specimens were developed in the early 1970s from analog bucket-brigade devices and quickly replaced the camera tubes. When they started conquering the consumer market in camcorders, the CCD sensors had reached an attractive price level accelerating their use in machine vision.

CMOS sensors: Research into CMOS sensors got impetus in the early 1990s. In 1993 the Jet Propulsion Laboratory (Pasadena, CA, U.S.A.) provided the US space agency NASA with a sensor having a resolution of 128×128 pixels being followed by the first one-megapixel sensor only three years later. Unlike the

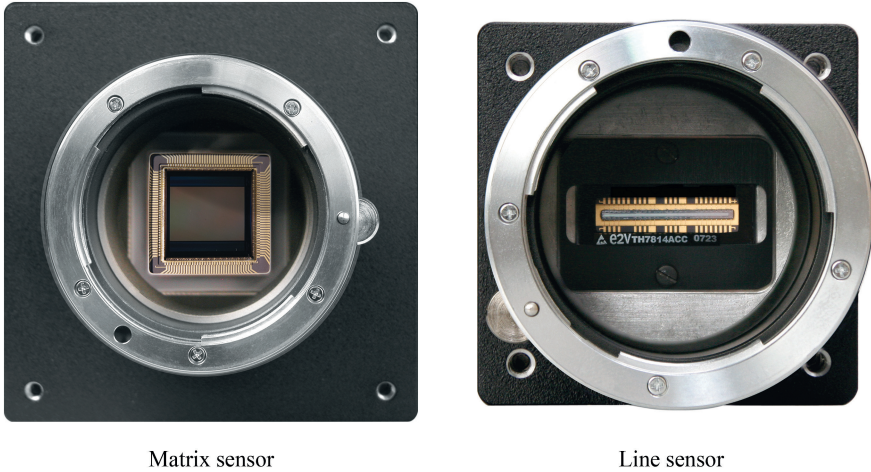


Fig. 8.1 Matrix and line sensors (courtesy of Basler AG (Ahrensburg, Germany))

manufacture of CCD sensors, the manufacture of CMOS chips and sensors is based on wide-spread standard technology. Therefore, the production of CMOS sensors is in principle more cost-efficient. More important for the sensor properties are the different ways in which image information is read out.

Basically all solid-state sensors operate according to the same principle:

1. The incident light in the form of a stream of photons generates free electrons in the light-sensitive part of the solid-state crystal.
2. The generated electrons are collected over a specific period of time, the exposure or integration time.
3. After the exposure time, the acquired charge is transferred pixel by pixel to the converter stage in the CCD sensor.
4. Finally, the acquired charge is converted into a proportional voltage that can be converted into a numeric value by an analog–digital converter.

CCD and CMOS sensors use photodiodes to convert incident photons into charge and collect them in the potential well. There the charge is stored during exposure time. The storage capacity depends primarily on the geometric pixel size and the build-up of the solid-state structures. It is denoted as *full well capacity* of the pixel in electrons (e^-). The rule of thumb for all sensors is important: the larger the pixel, the larger the full well capacity and thus the attainable range of dynamics and contrast.

A fundamental difference between CCD and CMOS sensors is in the read-out of the charges of the individual pixels. This has a significant effect on the sensor properties and will be discussed in detail in the following sections.

8.1.2 CCD Sensors

A solid-state sensor consists of light-sensitive photodiodes arranged in a matrix or, in the case of a line sensor, in a line. As soon as they are hit by light, charges are produced. Since digital cameras usually do not have mechanical apertures, the photodiodes are constantly exposed to light. To get usable images, the exposure time must have both a defined beginning and defined end. Some mechanisms are thus needed to limit the exposure time, i.e. the time the incident light contributes to the captured image. Since the light hitting the sensor during exposure is collected, it is called integration time.

Exposure time begins with the simultaneous removal of the charges in the photodiodes of all pixels. From this moment on, all generated charges are collected. Between the gaps of the photodiodes the vertical shift or transport registers are located, as can be seen schematically in Fig. 8.2. Because of these registers, this type is called interline transfer sensor. At the end of the exposure time, all charges representing the image information are simultaneously moved from the photodiodes of all pixels into the transport registers. Since the transport registers are mostly protected from incident light by a thin metal foil, no more charges are generated there.

Within the transport registers, the charges can be moved pixel-wise. The method used gave the sensor its name: charge coupled device. From the vertical transport registers, the charges are moved line by line into the horizontal transport register, from where they are read out pixel by pixel. This way, the image information captured by the sensor is read out as a sequence of charges of individual pixels.

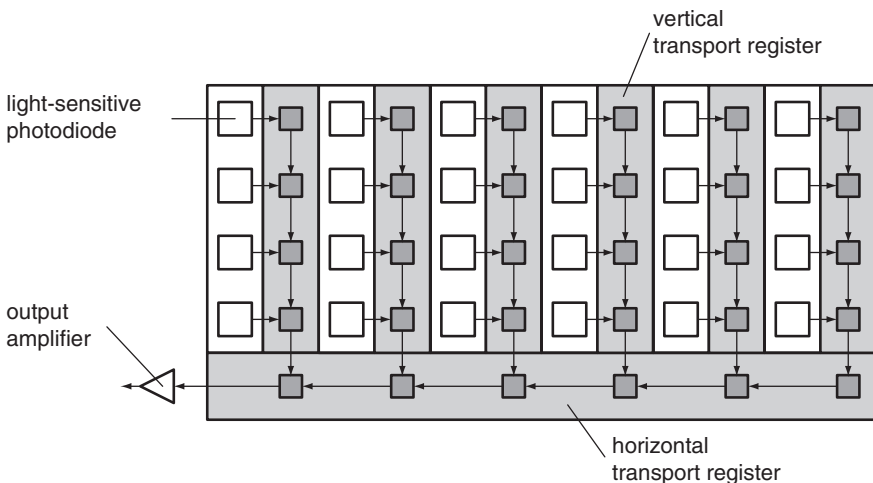


Fig. 8.2 Structure of an interline CCD sensor

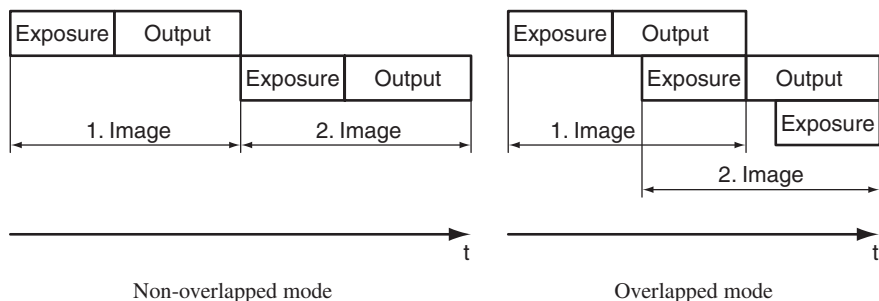


Fig. 8.3 Operation of a sensor in non-overlapped mode and overlapped mode

At the output of the horizontal transport register, the charge amplifier, called output amplifier, converts the charge packages of individual pixels into a voltage which is subsequently digitized. For the analog-to-digital conversion of the signal, correlated double sampling (CDS) technology is often used. The black values and then the white values of the signal are read. By subtracting both values in the analog-to-digital converter (ADC), a value proportional to the brightness is output. The advantage of this method is, for example, the suppression of low-frequency noises from the operating voltage of the output amplifier.

Since the integration and transport of charges takes place in separate areas, it is possible to expose the next image while the charges of the previous one are output. This is called overlapped mode and it can be used to capture sequential images more quickly than if exposure were to begin after the previous image has been completely output, as can be seen schematically in Fig. 8.3.

Besides interline-transfer architecture there are other types such as the full-frame sensors and frame transfer sensors. These architectures have no vertical transport registers, which is why they are more sensitive. However without mechanical shutter, such sensors can only capture static objects; hence, they are but rarely used in industrial image processing.

So far we have assumed that the entire image is output at once. This process is called progressive scan. There is another process that has been widely used for years: interlaced scanning. This process was borrowed from analog TV technology where image fields are used. Alternately, all even and all uneven lines are output. One of the disadvantages of this method is the highly reduced image rate because of the separate exposure of the fields. Distortions are another detriment that occur with moving scenes. Therefore, interlaced cameras are rarely used in machine vision today.

The speed used to move the charges of a sensor and read them out is subject to physical limitations. This limits the image rate, i.e. the number of images per second. Typical values for a readout channel are 40–60 MHz, i.e. 40–60 million pixels per second. To achieve higher image rates, especially for sensors with higher resolutions, architectures with multiple outputs, so-called taps, have been developed. The individual output amplifiers of these multi-tap sensors are used to read out the charges of a part of the sensor pixels in parallel. Thus a sensor with

four taps can read out four times faster than a sensor with just one output. The sensor depicted schematically in Fig. 8.4 has four taps. The charges of the pixels in the upper left quarter of the sensor are transported to the output amplifier on the top left; the charges of the pixels in the upper right quarter are transported to the amplifier on the top right etc. The challenge for manufacturers of multi-tap sensors is in balancing the analog-to-digital converters of the individual channels to ensure a homogeneous gray value level between the four parts of the image. Premium-quality industrial cameras can perform this balancing.

8.1.3 CMOS Sensors

The main difference between CMOS and CCD sensors is that CCD sensors—with the exception of multi-tap sensors—have only one charge amplifier for the entire sensor while with CMOS sensors each pixel has its own charge amplifier. These amplifiers convert the charges generated during exposure into voltage directly in the pixel. Because of this conversion, the CMOS sensors are also called active pixel sensors. The brightness information of each pixel is read out as voltage, not as charges as with CCD sensors. The circuitry for the readout of individual pixel voltages is depicted schematically in Fig. 8.5. From the vertical control the so-called select transistors of the pixels in one row are switched, thus selecting the image row for readout. Every active select transistor puts the voltage of its pixel on the corresponding column line. The horizontal control then switches one column line after the other to the output amplifier of the sensor which then reads out the voltages of the pixels in sequence.

As mentioned before, each pixel has its own charge amplifier for converting the entire collected charges into a voltage. These charge amplifiers differ from each other with regard to their amplification and offset thus the output voltages of

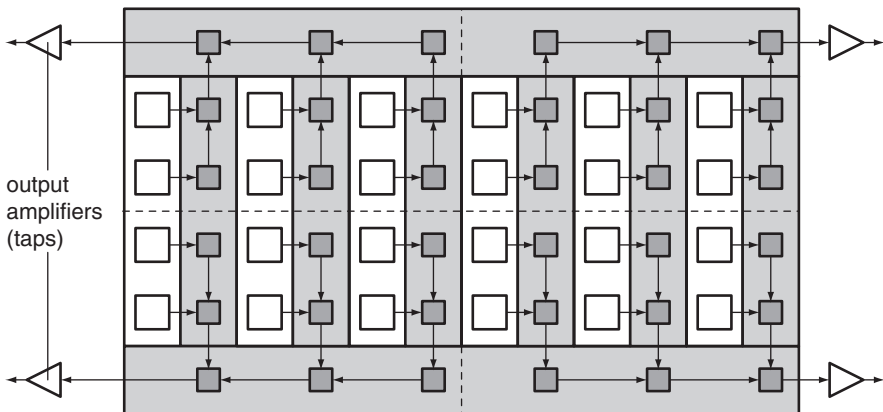


Fig. 8.4 Principle of a 4-tap sensor

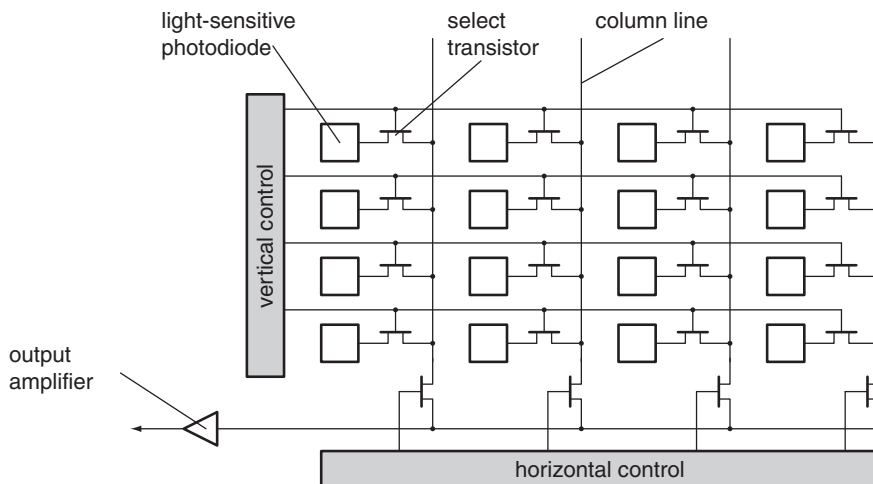


Fig. 8.5 Structure of a CMOS sensor

individual pixels with the same charge frequently vary. These differences are superimposed over the captured image as a stationary noise. This is called fixed pattern noise and is typical for CMOS sensors. By the correlated double sampling already mentioned with CCD sensors, fixed pattern noise can be reduced. However, this must be done for each pixel so the scope of the circuitry grows at the expense of the light-sensitive area. According to the above rule, this results in a low sensitivity of the CMOS sensor.

Besides the correlated double sampling, the circuitry of the individual pixels can also be extended with other functions. The analog–digital conversion often already takes place, for example, on the chip either using an analog-to-digital converter for the entire sensor or one for each column. Therefore, CMOS sensors often have a digital interface only. This allows downstream components to evaluate the output data directly. Other typical functions include integrated control of exposure time and readout of portions that are also called Regions of Interest just like the search areas in an image. If the sensor also contains functions for pre-processing or color conversion, this is called system on chip (SOC).

Exposure control: As with CCD sensors, CMOS sensors also generate charges as soon as light hits the light-sensitive areas of the pixels. Therefore, exposure control defining the beginning and end of exposure time is necessary for CMOS sensors, too. Two different methods are commonly employed:

Rolling shutter: The most common method is rolling shutter. As with CCD sensors, exposure begins with removal of already generated charges. With rolling shutter, this is not done simultaneously for the entire sensor, but row by row. At the same time the rows are also read out row by row thus ending the exposure time. The time offset between the beginnings of the exposure of two neighboring rows equals the timespan necessary for the readout of one row.

This method allows for the manufacture of simple and cost-efficient sensors with uniform exposure time for all pixels. That is why this type of sensor is frequently used in multimedia applications. However, the offset in exposure time for individual rows results in distortions for moving objects. Therefore, these sensors are only suitable in industrial image processing for capturing static objects.

Global shutter: Unlike rolling-shutter sensors, sensors with global shutter have a charge storage cell in the pixel. Thus it is possible to buffer the charges generated during exposure time until readout. This allows for the exposure time of all pixels of the sensor to begin and end simultaneously. The price is a more complex circuitry since buffering requires up to eight additional transistors per pixel resulting in a smaller light-sensitive area and thus lower sensitivity.

Currently there are but a few CMOS sensors with global shutters on the market. These are mostly sensors that have been manufactured exclusively for industrial image processing. Because of these small quantities, they are much more expensive than CMOS sensors with rolling shutter that are manufactured in large quantities for multimedia applications. Compared with CCD sensors, CMOS sensors with global shutters do not have a price advantage but the disadvantage that a lot of effort needs to be invested in correcting the inhomogeneity of the images. In recent years, CMOS sensors with global shutter have become available, exhibiting high light sensitivity, too, and with sufficient image homogeneity to be used in most areas of industrial image processing.

The readout speed of CMOS sensors depends on the number of readout channels. As with CCD sensors where readout speed can be accelerated with multi-tap architecture, the field of view of CMOS sensors can be divided into several areas, each of which has its own readout channel. With CMOS sensors the division into areas is done by pooling several columns. It's typical for this architecture that the readout speed increases when less rows are read out; the readout speed remains constant when less columns are read out.

8.1.4 Special Types

As mentioned before it is possible with CMOS sensors to integrate additional functionality simply and cost-efficiently into every pixel albeit at the cost of light-sensitive area. This allows for the implementation of special types for special applications. An example for a special type is the time-of-flight sensor (abbreviated TOF sensor) that captures a distance image in addition to the brightness image with the distance measurement being based on the propagation time of light, the "time of flight". Thus this sensor is also an example for a 3D sensor.

To measure the time of flight, a light source is necessary emitting light modulated with a sine curve that is synchronized with the sensor. The continuous increase and decrease of brightness in the light source can also be observed in the light reflected into the sensor by an object. The time of flight can be observed as the difference between the phases of emitted and received light. The sensor serves

as receiver synchronous with the modulation of the emitted light. During a quarter cycle, the charges are stored in one storage cell, then in the next. Since one period is too short to generate sufficient charges, the charges of several periods are collected in the four storage cells. At the end of the integration time the following can be computed from the charges collected in the storage cells:

- The mean values as a measure for the brightness of the object as with a normal camera sensor.
- Phase difference as a measure for the distance to the object.
- Amplitude as measure for the accuracy of the distance measurement.

Because of the extensive additional electronics, a pixel uses up a multiple of the area usually occupied by pixels of normal CMOS sensors. Thus the resolution of a TOF sensor is considerably lower than that of normal camera sensors. State of the art is currently resolutions of 200×200 . Thus the possibilities of application for TOF sensors are rather limited; however, this technology is an interesting example of special developments based on CMOS sensors.

Besides TOF sensors, there are a number of sensor systems also capable of gathering 3D information with the help of light section, stereoscopy and other methods. Just like TOF sensors, most of them can so far only be used for very limited areas of application. However, this is a highly dynamic market so some of these technologies will establish themselves in image processing over the course of the next years.

8.1.5 Color Sensors

While the brightness of gray level images can be described as a physical quantity of intensity and therefore can be used with infrared and ultraviolet light without problems, color is inseparably linked to human perception. Outside the wavelength range perceptible by the human eye between 380 and 780 nm, no colors are defined. The frequently encountered depiction of color with three channels in industrial image processing was derived from the human eye. The color-sensitive photo receptors, or cones, are divided into three different types in the human eye, each with its own spectral sensitivity so each type of cone reacts differently to light of the same wavelength. They are named according to the wavelength area in which the various types react most strongly:

- L-cones, for long-wave, red light
- M-cones, for the middle spectrum
- S-cones, for short-wave, blue light.

As can be seen from Fig. 8.6, the sensitivity of the L and M-cones is very similar while that of the S-cones is significantly different.

As mentioned above, colors are usually described with three channels in industrial image processing. Each channel represents one of the primary colors, or

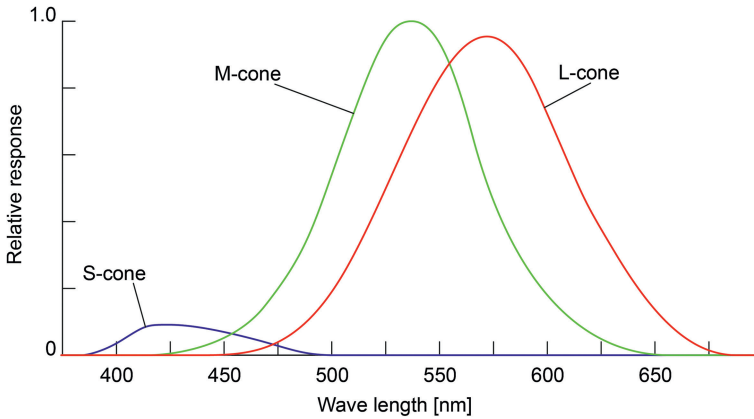


Fig. 8.6 Spectral sensitivity of cones in the human eye

primaries. With the weighted combination of the three primary colors, all colors perceptible by the sensor can be described.

Since an individual pixel can only absorb one brightness value, color measurements are not that easy. To gain color information, the incident light must be split by wavelength and be routed to different sensor elements. There are a number of methods for this:

Three-chip color cameras: With three-chip color cameras the light is divided into a red, green and blue component by a beam splitter on its way from the lens to the camera sensor. Each component goes to a different sensor chip where the corresponding red, green or blue image is generated. Thus all three color values are available for each pixel so the color resolution is excellent. A serious disadvantage of this camera is that it is rather big, heavy and expensive because of the beam separation and the required three sensors and the readout electronics. Another challenge is to make sure that the color channels are congruent on the three sensors. Since the incident angle of the beam splitter has an effect on the filter characteristics, such cameras can only be manufactured for small chips with low resolution.

Single-chip color camera with color filter mask: It is technically considerably easier to use only one chip for color image capture. A color filter array (abbreviated CFA) is directly applied to the sensor. The CFA makes sure that each pixel can only receive the brightness of one color channel. Since the neighboring pixels receive other color channels, the complete color information can be interpolated from the surrounding pixels. There are a number of different color filter arrays of which Fig. 8.7 shows a selection. The array used most frequently is the Bayer pattern¹ depicted on the left of Fig. 8.7. Single-chip color cameras have the

¹ Named after Bryce E. Bayer, who developed this pattern in 1976 at the Eastman Kodak Company (Rochester, NY, USA).

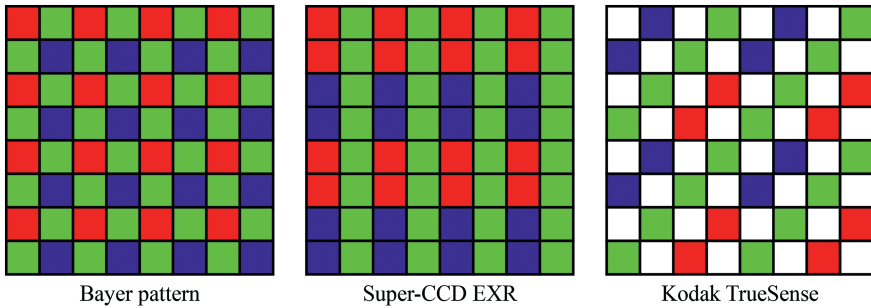


Fig. 8.7 Color filter arrays

disadvantage of having a lower spatial resolution since the color values of the neighboring pixels must be interpolated. Another disadvantage is the reduced sensitivity since transmission color filters always absorb part of the light from the wavelength area that they can nominally transmit.

Three-layer color sensor: A third option is the three-layer color sensor developed by Foveon Inc. (Santa Clara, CA, U.S.A.). With this sensor, three pixels are arranged on top of each other absorbing in each pixel layer the light of a different wavelength range and converting it into charges. The different-colored light reaches the next pixel layer, is absorbed and converted into charges, or passed on. So far this technology hasn't been accepted on the market.

Gamut: All colors absorbed by a color sensor are described by weighted addition of primaries. The colors of the primaries are derived from the spectral sensitivity of the silicon and the colors of the color filter matrix or the characteristics of the beam splitter. Because of these technical limitations, it's not possible to perceive all the colors a human can see with a color sensor. The range of the color spectrum a color sensor can perceive is called gamut. Different cameras may have different primaries thus differing in their gamut.

8.1.6 Properties of Sensors

Pixel size: The larger the pixels, the larger their light-sensitive area and thus their sensitivity. In addition, larger pixels can store a higher charge resulting in a larger dynamic range. Data sheets list this important sensor property frequently as pixel pitch, i.e. the distance between neighboring pixels. Typical pixel sizes for use in industrial image application are 4–7 μm since smaller pixel sizes generally do not make sense because of diffraction limitations of optical imaging. An exception are color CMOS sensors for the consumer market with pixel-sizes down to 2 μm . Here the neighboring pixels are included for calculating the RGB values of a pixel which puts spatial resolution at a minimum of 4 μm .

Table 8.1 Typical optical formats of solid-state sensors

Optical format	Width (mm)	Height (mm)	Diagonal (mm)
1/1"	12.8	9.6	16.0
2/3"	8.8	6.6	11.0
1/2"	6.4	4.8	8.0
1/3"	4.8	3.6	6.0

Resolution: With matrix sensors, the resolution denotes the number of pixels in horizontal and vertical direction, or the overall number of pixels on the chip in megapixel (1,000,000 pixels). With area sensors the aspect ratio is frequently 4:3, but there are also many other aspect ratios including square ones. Common resolutions in machine vision are between 640×480 , which corresponds to 0.3 megapixels, and $2,448 \times 2,050$, or 5 megapixels. For special applications, sensors with 16 megapixels or more are used.

With line sensors, the resolution denotes the number of pixels in the scan line. Since line-scan cameras can only capture one line at a time, and these lines are then put together to form an image, no number of rows can be given. Typical resolutions of currently available line-scan cameras are 1,024–8,192 pixels.

Optical format: The optical format of a sensor denotes the diagonal size of the image. This value is important when selecting a suitable lens since the screen image circle must at least be the same size as the diagonal of the sensor. The optical format is usually shown in inches with 1 inch corresponding to 16 mm. Originally this measure denoted the external diameter of the glass bulb of a camera tube. With a tube diameter of 25.4 mm (1 inch) the diagonal is 16 mm. The tubes have disappeared from the market, the measurement has stayed. Typical optical formats can be seen in Table 8.1.

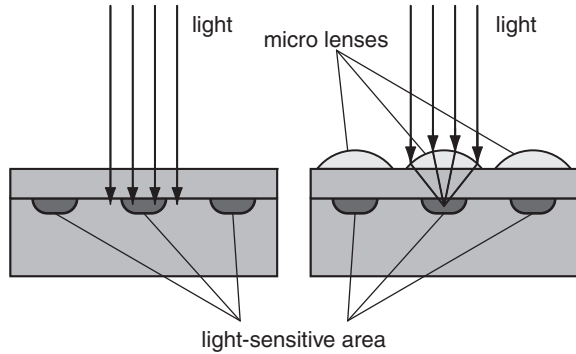
Quantum efficiency: Both CCD and CMOS sensors are based on the same principle that photons reaching a pixel generate charges. However, not every photon generates a charge carrier. For one, some photons hit light-insensitive areas of the pixel, but also for physical reasons not all photons hitting the light-sensitive area are converted into electrons. Quantum efficiency describes how good this conversion process is working by giving the ratio between electrons created and incident photons.

$$\mu = \frac{n_{\text{electrons}}}{n_{\text{photons}}} \quad (8.1)$$

Fill factor: A pixel consists of a light-sensitive and a light-insensitive area. The ratio of light-sensitive area to total area is called fill factor. The fill factor also contributes to the quantum efficiency.

Micro lenses: To improve quantum efficiency for solid-state sensors with a poor fill factor, micro lenses are frequently used. Micro lenses are applied as a matrix to the sensor chip so each micro lens corresponds to a pixel as can be seen in Fig. 8.8. This lens focuses the incident light on the light-sensitive area of the

Fig. 8.8 Micro lenses and their effect



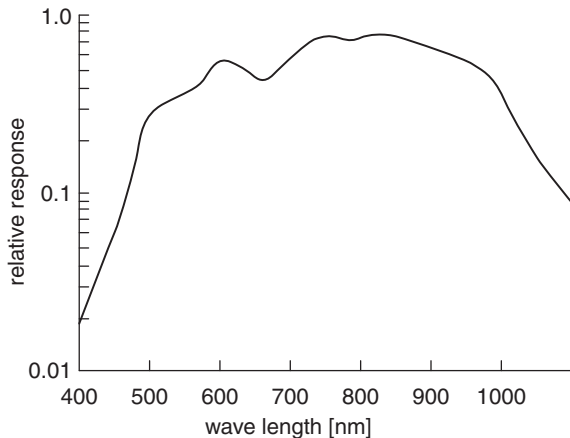
pixel. So light that would have hit light-insensitive areas is redirected to light-sensitive areas.

Spectral sensitivity: The quantum efficiency of solid-state sensors depends significantly on the wavelength of the incident light as can be seen in Fig. 8.9. The spectral sensitivity of solid-state sensors is limited by two effects:

Long-wave light: With increasing wavelength of the incident light, a photon penetrates the sensor material more deeply before it is absorbed and a charge is generated. Thus the probability of a charge being generated and stored within the photodiode decreases with increasing wavelength. For silicon—which is what most solid-state sensors are made of—photons must have a wavelength of less than 1,125 nm to emit electrons. Silicon-based sensors are insensitive to light with longer wavelengths.

Short-wave light: The opposite is true for short-wave light. Short-wave light is absorbed close to the sensor surface and generates charges there, which are therefore frequently too far from the charge storage so they cannot be stored. So sensitivity also declines in the short-wave area of the spectrum.

Fig. 8.9 Spectral sensitivity of a CCD sensor



To describe spectral sensitivity, data sheets usually list the relative response factor (abbreviated $R/R_{\max}$), which gives the ratio between quantum efficiency at a certain wavelength and the maximum quantum efficiency.

Linearity: The charge of a pixel increases linearly with the number of incident photons until full well capacity is reached. Minor non-linearities can occur in the charge amplifier but in general they are only important for very precise color measuring tasks. Linearity describes how well the output signal of the camera increases linearly with the number of incident photons.

Noise: The image information is always superimposed by random noise that has various causes. A number of noise types are generated in the image capturing sensor, some noise is generated in the camera electronics which we will discuss in Sect. 8.2.3.

Photon noise: Photon noise is an inevitable quantum effect independent of the sensor quality. When the pixels are filled by two-digit percentage points of their full well capacity, this is the strongest source of noise.

Fixed pattern noise: Fixed pattern noise, described in Sect. 8.1.3., mostly occurs in CMOS sensors.

Image capturing effects: During image capturing with solid-state sensors various effects occur distorting the original scene. Typical faults are:

Smear: Smear only occurs with CCD sensors. Since transport registers can never be completely shielded against light, charges are created that are not deleted at the beginning of the exposure time. Upon readout of the pixels, these charges from the transport registers are added to the charges from the pixels. The higher the intensity of light and the shorter the exposure time, the stronger will be the smearing effect. As a remedy against this issue, it is recommended to switch off the light source during read out of charges, preferably by using strobed lighting.

Blooming: If too much light falls onto a pixel, more charges are generated than can be stored in the pixel. The charges that cannot be stored flow into neighboring pixels thus enhancing their brightness. This effect is called blooming. Since the charges cannot lightly cross the transport registers, blooming usually spreads vertically along the columns. Anti-blooming drains are used to try to drain excessive charges in CCD sensors to avoid blooming. CMOS sensors frequently contain anti-blooming functions in individual pixels and are therefore less susceptible to blooming than CCD sensors.

In conclusion of this chapter about solid-state sensors, the most important advantages and disadvantages of CCD and CMOS sensors are summarized. First the advantages of CCD over CMOS sensors:

- little noise, especially low fixed-pattern noise
- low dark current
- global shutter.

In contrast, there are a number of advantages of CMOS over CCD sensors:

- low manufacturing cost for mass-market sensors
- high degree of integration

- low power consumption
- image sections can be read out
- higher readout speed.

8.2 Digital Cameras

Unlike the analog data processing of older cameras, modern digital cameras, as can be seen in Fig. 8.10, convert all signals of the image capturing sensor into digital values. The digital values are prepared in the camera for transferal over a standardized, digital data interface. Thus the lossless transfer of image data from the sensor to the image processing computer is ensured.

A number of functions and properties contribute to the performance of a digital camera. This section will begin with introducing several fundamental functions of all cameras and then functions specific to color cameras. Then we will present characteristic values for the description of relevant camera properties. In conclusion we will discuss the special conditions cameras in industrial environments are exposed to.

8.2.1 Control of Image Capture

Digital cameras have a processor, frequently a field programmable gate array (abbreviated FPGA). This processor controls image capture and processes the image data before transferal to the image processing computer. Control of the image capture is extremely important to get an image that is sufficiently bright and rich in contrast. Control of the image capturing process is influenced by a number of parameters.



Fig. 8.10 Modern digital cameras

Exposure time: The most important parameter in image capturing is exposure time, which is used to control the time span over which charges are collected in the pixels. With increasing exposure time, the captured image gets brighter, so exposure time should be long enough to get a sufficiently bright image. Some applications, however, won't allow for a sufficiently long exposure time. With moving parts, for example, one has to choose a short exposure time to minimize motion blur.

Signal amplification (gain): If the application doesn't allow reaching sufficient image brightness by prolonging exposure time or optimizing illumination and optics, the signal generated in the sensor can be artificially amplified by the analog-to-digital converter. However, along with the image signal, the noise is amplified as well so a high gain frequently leads to significant noise. Increasing the image brightness with a higher gain should only be used if the means of illumination, lens and exposure time have been exhausted.

Automatic brightness control: Usually, exposure time and amplification are fixed values. If the object illumination changes, the image will be brighter or darker. To counteract this change in brightness, many cameras have automatic exposure control (AEC) and automatic gain control (AGC). In industrial image processing these two functions are hardly ever used since illumination is usually stable and changes in exposure time and gain cannot be foreseen.

Control of external illumination: We cannot stress enough that a good illumination adjusted to the environment and work piece is of the utmost importance to capture an image that can be evaluated at all. Industrial image processing frequently uses strobes. Digital cameras usually have special digital outputs that can be used to switch on illumination during exposure time.

Synchronizing image capture: An image has to be captured at the precise moment the inspection part is in the field of view of the camera. Therefore, image capture must be triggered at the right moment, which is why the signal actuating image capturing is called a trigger. The trigger can come from various sources. The two triggers typical for industrial image processing are hardware trigger and software trigger.

Hardware trigger: With the hardware trigger, the camera is directly connected to the signal source by a cable. The signal is usually transmitted as a change of a voltage level. Light barriers are typical signal sources. When the inspection part breaks the light barrier, the output level of the barrier rises causing the camera to start image capture. Hardware triggers are usually employed when precise synchronization of image capturing is required. Usually, the hardware trigger works with a latency of only a few microseconds.

Software trigger: The software trigger signal is generated by the image processing software on the evaluation computer by a method call and transferred via digital interface to the camera. Because of the longer signal path and the specific properties of the operating system on the PC, latency periods of several milliseconds can occur with the software trigger.

Frame rate: The number of images being captured within one second is called frame rate, derived from the term video frame. The frame rate depends significantly on the exposure time, the time for the readout of the image and the time for

transferring the image from the camera to the image processing computer. All three time periods are the greater, the more pixels are exposed, read out and transferred. It is therefore obvious that the frame rate can be increased by reducing the number of pixels. Two methods to reduce the number of pixels are partial scan and binning. Besides reducing the number of pixels, the frame rate can be increased by exposing and reading out the image data simultaneously as described in Sect. 8.1.2.

Partial Scan: Some sensors allow for only a limited portion of the pixels to be read out. This function is called partial scan and the read out part of the sensor is called Region of Interest like the search area. Thus partial scan can be used to increase the frame rate when all the information relevant for evaluation is contained within a limited image section.

Binning: With binning the charges of neighboring pixels are concentrated in one “super pixel”. This can take place horizontally, vertically or in both directions. By binning, the number of pixels is halved or quartered thus increasing the frame rate. With some sensor types, the light-sensitive area of the super pixel is twice respectively four times as large as that of the individual pixels. Because of the thus achieved higher light sensitivity of the super pixel, the exposure time can also be shortened if necessary.

8.2.2 Capturing Color Images

We explained in Sect. 8.1.5 the architecture of solid-state sensors for capturing color images. In this section we will discuss how a color image is generated from the sensor’s raw data. The necessary process comprises demosaicing, color calibration, white balance, and output in a usable format.

Demosaicing: As described in Sect. 8.1.5, in industrial image processing mostly single-chip cameras with overlaid CFA are used. Since with this type of camera only the value of one of the usually three color channels is available for each pixel, the values of the other two channels must be interpolated from the neighboring pixels. The interpolation of color values is called demosaicing and is usually done by the camera’s processor. With the Bayer pattern (on the left in Fig. 8.7), which is the de facto standard for color filter arrays, half of the pixels are green and a quarter each is red and blue respectively. Thus for half the pixels the green value needs to be interpolated and for three quarter of the pixels the red value and blue value respectively need to be interpolated. There are a number of calculation methods for this interpolation. The most simple merely repeats the color values of the neighborhood while the more complex methods look at a neighborhood of up to 7×7 pixels around the point and undertake elaborate calculations to interpolate the color values. The simple approaches have the advantage of fast execution; however, frequently color fringes can be observed along edges, i.e. stripes of a different color running parallel to the edge. The more complex methods generate images with sharp edges without color fringes and a

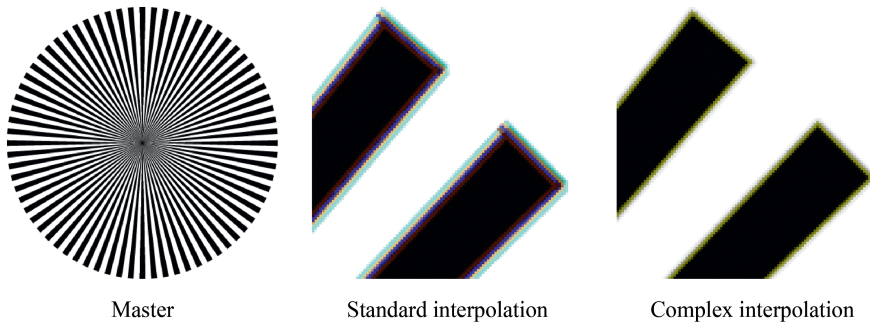


Fig. 8.11 Comparison of two demosaicing methods

more accurate color reproduction as can be seen in Fig. 8.11. In the end, the selection of an interpolation method constitutes a compromise between frame rate and image quality, and it depends on the inspection task, which of the two is more important for the application.

Color calibration: The raw data of a digital color camera and the RGB data interpolated from them are device-specific since they depend on the spectral sensitivity of the solid-state sensor and the colors of the color filter array. If there are no great demands on the quality of color acquisition, e.g. with a simple classification of colored objects, this device-specific data is completely sufficient.

If the camera is to measure color precisely, or the camera is to be used for visualization tasks such as in medical technology, microscopy or printing, the device-specific color data needs to be transformed into a standardized color space. This is called the color calibration of the camera. Common standardized color spaces include:

- CIE XYZ color space comprises all visible colors and is based on the colors red, green and blue. It was developed in the 1930s by the International Commission on Illumination (French: Commission Internationale de l'Éclairage, abbreviated CIE).
- CIE Lab color space was also defined by CIE for the color-metric evaluation of colors.
- sRGB color space is adjusted for the display on computer screens.

White balance: The light being captured by a color camera is determined by the object color and the light color of the illumination. To capture comparable color values, the camera needs to be adjusted to the prevailing illumination conditions. This adjustment is called white balance or color balance. Here each channel is amplified with various factors to compensate for the imbalance between the channels.

The spectral distribution of the light from a light source is called color temperature and is stated in the unit of the kelvin (abbreviated K). The given temperature is the temperature of a body radiating light as thermal radiation with the same spectral composition. Common color temperatures are 5,000 and 6,500 K for daylight

(abbreviated D50 and D65), and 2,856 K (also called standard illuminant B) for the light of light bulbs.

White balance has to take place with the same illumination used during inspection since the camera has to be adjusted to these colors. The white balance function optimizes the gain factors of each channel in such a way that as many pixels as possible have identical red, green and blue values, which means they are gray. Therefore, the camera should be capturing gray levels for white balance. Optimal white balance is achieved when the camera only captures a gray wedge, and when no overexposure because of a too bright sample or too long exposure time occurs.

Output formats: The data of a color image can be transferred to the image processing PC using various output formats. There are two types of output formats:

- Raw data
- Computed formats

Raw data: The original data from the sensor without demosaicing in the camera is called raw data. The PC then has to do the color interpolation, which, on the one hand, puts quite a load on the PC; on the other hand, it allows for a color interpolation especially suited to the application at hand.

Computed formats: Typical computed formats are the RGB and YUV format. In image processing the RGB format is very common since it closely fits the hardware. Cameras as well as most displays operate based on RGB. The YUV format is mostly used for visualization applications since the amount of data can be reduced in such a way that the lessening of image quality is hardly noticeable to the human eye. However, in industrial image processing the YUV format plays only a minor role.

8.2.3 Characteristic Values of Digital Cameras

The quality and performance of a camera can be described using a number of characteristic values such as spectral sensitivity (cf. Sect. 8.1.6), the signal-to-noise ratio (SNR), dynamics or linearity. These parameters and the measuring methods to determine them are defined by the EMVA 1288 standard of the European Machine Vision Association (EMVA). This standard is continuously developed further by a committee consisting of camera and sensor manufacturers, distributors and researchers to take into account the latest developments in camera and sensor technology. Camera quality heavily depends on the camera sensor, so many characteristic values such as linearity or homogeneity refer mainly to sensor properties we discussed in Sect. 8.1.

Signal-to-noise ratio: One of the most important quality characteristics of a digital camera is signal-to-noise ratio. The image quality is the better the higher the signal-to-noise ratio. One of the noise sources in the camera as a whole is photon noise which is generated in the sensor and which we discussed in Sect. 8.1.6. Nevertheless, there are other kinds of noise that are not only caused by the sensor:

Thermal noise: Even if no light hits the pixels, thermal effects in the semiconductor generate charge carriers. This charge carrier generation occurs in all electronic components of a digital camera, i.e. in the pixels, the transport registers, the readout and amplifier electronics, and is called dark current. The variance of the dark current is called thermal noise and differs from pixel to pixel. Only for exposure times of more than half a second does it contribute significantly to the overall noise.

Amplifier noise: The amplifier or read-out noise is generated when converting charges into a digital signal and is therefore heavily influenced by the quality of the camera electronics. It is only dominant with very low levels and normal exposure times since photon noise and thermal noise are smaller under these conditions.

Dynamic range: Another important characteristic value for camera quality is the dynamic range. This is the ratio of the largest brightness value and the smallest brightness value that differs from noise. Typical values of high-quality cameras are higher than 50 dB.

Linearity: Linearity depends significantly on the sensor and has been discussed in Sect. 8.1.6.

Homogeneity: Homogeneity is a characteristic value for the fixed pattern noise of CMOS sensors described in Sects. 8.1.2 and 8.1.6.

8.2.4 Operating Conditions in Industrial Environments

Temperature: The operating and ambient temperature play a vital role in the quality of image capturing and the durability of the camera used. Part of camera noise is dependent on sensor temperature and increases with it. Heat sources are the camera itself with power dissipation as an indicator for the heat produced, and other external sources such as illumination or other electrical consumers in the vicinity of the camera. One indicator of high-quality industrial cameras is therefore low power dissipation. Because of the limited space in industrial visual inspection lines, cameras with small casings are necessary which results in poor heat dissipation. With these types of cameras it is important to ensure good thermal coupling of the camera casing and the fixings.

Ingress protection: In many industrial processes cameras are operated in unfavorable environmental conditions such as humidity or dust. The camera's image quality is considerably worsened if dust or water ingress between the sensor and the lens. To avoid such problems and ensure high reliability even under adverse environmental conditions, digital cameras can be equipped with special protection against contaminants and humidity. This usually takes the form of an additional casing around the camera and the lens as can be seen in Fig. 8.12. In applications with special hygienic requirements as in the food industry, corrosion-resistant material such as stainless steel is used for the casings. The degree of protection is indicated by the IP code as described in the IEC 60529 standard. In

Fig. 8.12 Digital cameras with IP67 protective casing (courtesy of Baumer Optronic (Radeberg, Germany))



the designation IP67, the six indicates the casing is dust tight and there is complete protection against contact (e.g. with a wire). With the seven the manufacturer guarantees that the camera may be temporarily immersed in water without being damaged.

Power supply: For digital cameras to operate properly, a steady power supply and switch signals with good signal quality are essential. Unreliable voltage and interference sources regularly lead to erratic behavior of cameras in industrial image processing. The requirements on all electric components with regard to electric level, electromagnetic compatibility (EMC), and electrical safety are described in the standard on Programmable Logic Controllers (part 2) IEC 61131-2:2007.

Components: Besides suitable cameras, other components such as cables, interface cards for the image processing PC, power supply etc. play a vital role for the reliable use of digital cameras in industrial image processing. However, selecting individual components that are basically suitable does not suffice, they must also harmonize with each other.

8.3 Image Data Transfer

Up until the end of the 1990s, machine vision relied primarily on analog cameras that used the well-known TV analog transfer standards for image transfer. Special plug-in cards, known as frame grabbers, were used to digitize analog camera signals, which were then made available in the computer's memory as digital images for further processing. In the mid-1990s the first digital transfer methods emerged that allowed for the transfer of image data digitized in the camera as digital signals loss-free and with a high data rate directly to the PC. By now, digital data transfer prevails in new applications, almost completely eliminating analog transfer which is why we will only discuss digital image transfer here.

The first fully digital cameras to emerge in the market had proprietary interfaces such as LVDS or serial interfaces like HotLinkTM. From the LVDS interface CameraLink[®] was derived, the first interface standard designed especially for

Table 8.2 Overview of common digital camera interfaces

	CameraLink [®]	FireWire [®]	USB	Gigabit Ethernet
Transfer rate	6.8 Gbit/s	a: 400 Mbit/s b: 800 Mbit/s	2.0: 480 Mbit/s 3.0: 5.0 Gbit/s	1.0 Gbit/s
Cable length	10 m	4.5 m	5 m	100 m
Power supply	11–13 V, 0.4 A ^a	8–30 V, max. 15 W	2.0: 5 V, 0.5 A 3.0: 5 V, 0.9 A	37–57 V, max. 30 W ^b
Connection	Point to point	Bus	Bus	Network
PC interface	Frame grabber	Partially available	2.0: All PCs 3.0: New PCs	All PCs
Real-time capability	Excellent	Good	Reasonable ^c	Good
CPU load	Very low	Low	Medium ^c	Low to medium
Availability of components	Medium	Medium	2.0: Excellent 3.0: Medium	Excellent
Cost	High	Medium	2.0: Low 3.0: Medium	Medium
Industry suitability	Good to medium	Good	Reasonable ^c	Good

^a only with Power over CameraLink[®] (PoCL)

^b only with Power over Ethernet (PoE)

^c so far, there are no empirical values for USB 3.0

image data transfer, with regulations for cameras, cables and frame grabbers. While CameraLink[®] was especially developed with image processing in mind, a number of digital interfaces from the consumer market are also used in industrial image processing. The first was the IEEE1394 standard, better known as FireWire[®], followed by Universal Serial Bus (USB), and finally Gigabit-Ethernet (GigE). These interfaces will be discussed in detail in the following. Table 8.2 provides a quick overview of the interface properties.

8.3.1 CameraLink[®]

The CameraLink[®] standard was the first interface especially developed for transferring image data in industrial image processing. Development and standardization were led by the American Automated Imaging Association (AIA). Although this standard has been developed in the 1990s, because of its high transmission speed, it is still the interface of choice in high-speed applications.

Since CameraLink[®] is not a standard PC interface, one needs special plug-in cards, CameraLink[®] frame grabbers, to transmit image data to the image processing computer. The advantage in the use of frame grabbers is that the

transmission of image data is independent of the latency periods of the operating system. In addition, the CPU load of the image processing computer is reduced since the frame grabber manages the reception of the image data and transmits the image data directly to the computer's RAM via Direct Memory Access (DMA). However, CameraLink[®] frame grabbers are expensive specialized hardware. Frequently, besides image transmission proper, frame grabbers are able to perform image preprocessing functions such as filtering. This also reduces the load on the image processing computer; however, the application is then dependent on the hardware employed.

Data transmission according to the CameraLink[®] standard is based on channel-link technology developed by National Semiconductor (Santa Clara, CA, U.S.A.). With a clock rate of up to 85 MHz, a channel-link is able to transfer 28 bit per clock. These 28 bits are collected in three ports with eight bit each. Three of the remaining four bits are used to indicate that the image transfer (Frame valid), the line transfer (Line valid) and the data transfer (Data valid) have been completed. For various band width requirements so-called configurations have been defined that are listed in Table 8.3. For each configuration the allocation of ports to image data is defined for various bit depths and pixel formats.

Besides transferring data from the camera to the image processing computer, CameraLink[®] provides two serial transfer channels for communication between computer and camera. Thus it is possible to configure the camera without additional hardware. Since the format of the configuration data is not standardized, manufacturer-specific programs are necessary for configuration. Furthermore, four control bits are available for, e.g. triggering the camera via the frame grabber. These control bits are called CC1, CC2, CC3 and CC4.

One disadvantage of CameraLink[®] compared with bus systems like FireWire[®] or USB was that it wasn't possible to supply the camera with power over the data cable thus an additional cable was always needed for power supply. By introducing Power over CameraLink[®] (PoCL) in version 1.2, i.e. power supply via data cable, this was remedied. The standard has the camera supplied with a voltage of 11–13 V with a maximum current of 0.4 A.

The next planned extension of the standard is increasing the data transfer rate to 6.8 Gbit/s via ten ports.

As can be seen from Table 8.3, one or two CameraLink[®] cables are necessary for the connection between camera and frame grabber. Since the port of a frame grabber can only be used by one camera, it is a point-to-point architecture. This way there can be no bandwidth contention since the full bandwidth is always available to the camera.

Table 8.3 CameraLink[®] configurations

Configuration	Channel links	Ports	Data rate (Gbit/s)	Cable number
Base	1	3	2.04	1
Medium	2	6	4.08	2
Full	3	8	5.44	2

Since data is transmitted in parallel, CameraLink[®] cables require rather many wires thus necessitating high-quality cables and connectors. This is mirrored in the components' price and also limits the cable length to ten meters; however, new cables with less attenuation allow for greater lengths. Originally, the standard defined only large 26-pin connectors in Miniature-Delta-Ribbon (MDR) format. Following a general trend to miniaturization, the standard was extended to Shrunk-Delta-Ribbon format plugs.

8.3.2 FireWire[®]

FireWire[®] is a serial bus developed by Apple Inc. (Cupertino, CA, U.S.A.) and Sony Corporation (Tokyo, Japan) to connect multimedia devices like camcorders, VCRs, digital cameras or external hard-disks to a computer. The first specification was in 1995 as IEEE1394a-1995 standard. The standard defines serial data transfer with a maximum transfer rate of 400 Mbit/s. In 2002 the extended IEEE1394b standard with transfer rates of 800, 1,600 and 3,200 Mbit/s was published. While the first chips for transfer rates of 800 Mbit/s were already available in 2003, no implementations for 1,600 and 3,200 Mbit/s can be found on the market so far. Figure 8.13 shows a digital camera with an IEEE1394b interface.

High data transfer rate and simple handling quickly established FireWire[®] as interface for image transfer of digital cameras in machine vision. Other reasons are high resistance to interference, plug-and-play capability and the ability to supply connected cameras with sufficient power to operate high-performance cameras with large image formats without additional power supply cables. During the past years, FireWire[®] has proved in many applications in industrial image processing to be a stable and extremely reliable interface. The most important features of FireWire[®] are:

- Plug-and-Play capability: plugging or unplugging cameras during operation is detected.
- Power supply: common PC cards supply cameras with 12 V and up to 500–750 mA electrical power.

Fig. 8.13 IEEE1394b camera (FireWire interface on top, bottom right trigger input, courtesy of Baumer Optronic (Radeberg, Germany))



- Recognition of transfer rate: the bus master recognizes the maximum transfer rate of the cameras automatically.
- Cascading several devices is possible (daisy chain).
- Downward compatibility: cameras supporting the new IEEE1394b standard are downward compatible to the IEEE1394a standard.

It could be problematic to use FireWire® further in the future since USB has established itself as the main PC peripheral interface, which is why FireWire® interfaces are no longer standard PC interfaces.

Isochronous image data transfer: For transferring digital images via FireWire®, the option of isochronous image data transfer is provided. The bus master generates a signal every 125 μ s to which all cameras connected to the FireWire® bus may attach their image data. The size of the data blocks that can be attached can be configured and determines the transfer rate. Table 8.4 shows typical package sizes and resulting transfer rates. Isochronous image data transfer does not allow for the recognition of transfer errors so transferred images may contain wrong pixel values. Especially in interference-prone environments, which are the rule in industrial image processing, this can present a problem.

DCAM standard: based on isochronous image data transfer, the 1394-based Digital Camera Specification (DCAM standard, known in the U.S. as Instrumentation and Industrial Digital Camera standard, IIDC standard) was developed defining uniform commands and formats for the control and data transfer of digital cameras. This allows for controlling cameras of various manufacturers with one implementation. However, the functional range defined by the standard is small.

Asynchronous image data transfer: Unlike isochronous image data transfer, asynchronous transfer offers secure transfer since every data package contains, among others, a check sum that enables the recipient to verify that the data is undamaged. If the recipient notices an error, the data package is requested again. Since undamaged data packages are acknowledged by the recipient, the camera can transfer the package again if the acknowledgment is not received. With asynchronous image data transfer higher data rates can be achieved than with isochronous image data transfer, as can be seen from Table 8.5.

A number of additional components are required when using FireWire® cameras, such as:

- Interface cards; for connecting FireWire® cameras to the image processing computer if the motherboard does not offer a FireWire® interface.

Table 8.4 Package size and maximum data rate for isochronous image data transfer

Transfer rate (Mbit/s)	Package size (byte)	Maximum data rate (Mbyte/s)
100	1,024	08
200	2,048	16
400	4,096	32
800	8,192	64

Table 8.5 Package size and maximum data rate for asynchronous image data transfer

Transfer rate (Mbit/s)	Package size (byte)	Maximum data rate (Mbyte/s)
400	2,048	40
800	4,096	80

- Hubs; allow the connection of several FireWire® cameras and can be used as repeaters to extend cable length.
- Cable; the IEEE1394 standard specifies a cable length of up to 4.5 m. However, high-quality FireWire® cables for industrial use allow reliable data transfer for up to ten meters even in an industrial environment.

Unfortunately, not all available components are suitable for use in industrial image processing since they either do not operate reliably or do not fulfill the other technical requirements. In general, the manufacturers of FireWire® cameras provide compatibility lists that detail the components that are suitable for use in industrial image processing.

8.3.3 USB

The Universal Serial Bus (USB) is probably the most important interface for the connection between PCs and peripheral equipment in the last decade. It was developed in the 1990s by Intel Corporation (Santa Clara, CA, U.S.A.). In 2000 the 2.0 version was published allowing for a data rate of 480 Mbit/s. Since this provided a sufficient data rate for transferring image data, USB was also used in industrial image processing. An advantage was certainly the wide-spread use of USB in the consumer market, furthermore plug-and-play capability and the possibility to supply cameras with power via the data cable.

Especially the lower EMI resistance when compared with FireWire®, the higher CPU load of the image processing computer and the reduced real-time capabilities have prevented USB so far from asserting itself as image data interface in industrial image processing. Thus the standardization of image data transfer via USB 2.0 has not happened so far.

In 2008 the 3.0 version of the USB standard was published enabling a very high data rate of 5 Gbit/s (Table 8.6). On new personal computers, USB 3.0 has begun to establish itself as a standard interface, so one can count on a high availability of components shortly. Currently, several camera manufacturers are working on introducing digital cameras with USB 3.0 interface to the market. In January 2013, the USB3 Vision™ standard was published. Whether USB can establish itself thus in industrial image processing, remains to be seen.

Table 8.6 Data rates of USB

Specification	Speed class	Data rate
1.1	Low speed	1.5 Mbit/s
	Full speed	12 Mbit/s
2.0	High speed	480 Mbit/s
3.0	Super speed	5 Gbit/s

8.3.4 Gigabit Ethernet

Ethernet was developed in the 1980s and is now established as the de facto standard for local computer networks. The third generation is called Gigabit Ethernet (GigE) or 1000BASE-X because of the 1 Gbit/s data rate. It is this data rate that makes GigE interesting for transferring digital image data.

GigE Vision: The AIA, which has already developed the CameraLink® standards, also developed the GigE Vision standard for image data transfer of digital cameras in industrial image processing. As can be deduced from the name Gigabit Ethernet is used as transfer medium. The standard uses some protocols common in IT as can be seen in Table 8.7.

The GigE Vision standard defines a protocol that may be used to implement a number of fundamental functions such as:

- Recognition of GigE Vision cameras in the network.
- Monitoring of connected devices.
- Sending and receiving image and control data.
- Fragmentation of large amounts of data.
- Error recognition and correction.

These functions are included in the various elements that make up the GigE Vision standard:

- GigE Device Discovery Protocol, the mechanism for finding GigE Vision cameras in a network and integrating them.

Table 8.7 Protocol layers of GigE Vision

Layer of OSI reference model	Protocol	Standard	Task
Application (7)	Pattern set		Processing of image data
Session (5)	GVCP/GVSP	GigE Vision	Image transfer and camera control
Transport (4)	User datagram protocol (UDP)	RFC768	Data exchange between end users
Network (3)	Internet protocol (IP)	RFC791	Addressing of data
Datalink (2)	Ethernet frame	IEEE802.3	Synchronization between sender and recipient
Physical (1)	Hardware	IEEE802.3	Transfer

- GigE Vision Control Protocol, for controlling a camera, the exchange of messages, the synchronization of camera and image processing computer, and sending of asynchronous messages by the camera.
- GigE Vision Streaming Protocol, the description of image data streams and data types.

Furthermore, GigE Vision defines some camera registers (so-called bootstrap registers) and their content necessary for communication between image processing computer and camera.

Network topology: The most simple way to connect a camera to an image processing computer is a point-to-point connection. Here the camera is connected directly to a PC Ethernet port so the full bandwidth is available for transferring image data.

If several cameras are connected to a PC via point-to-point connections, several Ethernet ports are necessary on the PC. To reduce this number, a network can be established connecting several cameras to a switch and this switch to a PC Ethernet port. Then the image data of all cameras must be transferred via the one connection between switch and PC so the cameras have to share the bandwidth. Thus the data bandwidth available to each camera is reduced when employing switches. The network of cameras and PC also needs to be managed so the data bandwidth available can be used in an optimal way allowing cameras and PC to communicate without difficulty. With such a network it is also possible to transfer one camera's image data to several PCs. Thus the image can be transferred to the image processing computer for evaluation and simultaneously to the HMI of the line for display.

Components: Ethernet was developed for the exchange of data between computers thus transmitting image data makes unusual demands on the transmission medium. For using Gigabit Ethernet as an interface in industrial image processing, one has to make sure that all components, especially the PC Ethernet adapter and any switches, fulfill the special requirements. As with FireWire[®], the manufacturers of Gigabit Ethernet cameras provide lists with suitable components.

Registered connectors: Registered connectors are the common eight-pin modular plugs used for Ethernet with an assignment according to RJ-45 which are usually complemented with screws for secure fastening for industrial use, as can

Fig. 8.14 Gigabit Ethernet camera (Ethernet connector on top, bottom left power supply, bottom right trigger input; courtesy of Baumer Optronic (Radeberg, Germany))



be seen in Fig. 8.14. These connectors are not suitable for higher ingress protection levels which is why M12 connectors are used quite often. There are already plugs for data connections up to 10 Gbit/s (10G Ethernet) available.

Cables: Today, twisted-pair cables are used exclusively for the connection of the cameras with the image processing computer or switch while glass fiber cables commonly used in networking are not used. Cable quality is an essential aspect for the reliable operation of a GigE Vision system in industrial image processing. In an industrial environment, at least Category 6 (Cat 6) cables should be used for sufficient shielding against interference. When installing cables in a cable carrier or robot, the type and quality of the cables is of great importance for permanently ensuring reliable data transfer.

Power over Ethernet: Just like CameraLink[®], Ethernet per se does not have the option to supply cameras with power via the data cable. Like CameraLink[®], the Ethernet standard was extended by the so-called Power over Ethernet (PoE). Direct current is injected into the differential pairs of twisted-pair cables so cameras can be supplied with power. The characteristic values of PoE power supply are:

- Supply voltage: 44–57 V.
- Maximum supply current: 350 mA.
- Maximum loop impedance: 20 Ω .
- Minimum available power at receiver-end: 12.95 W.

The power is injected into the network cables by midspan devices or power injectors that are installed between the image processing computer and the cameras. Instead of these devices one can also use PoE capable switches to supply the power. Today, there are already network cards available providing PoE supply. This can be used to enable a solution using only one cable without additional components.

Increasing transmission bandwidth: With high-resolution sensors or high data rates, Gigabit Ethernet reaches the limits of transmission bandwidth at 1 Gbit/s. To increase this, one can switch to the next Ethernet generation with 10 Gbit/s transmission bandwidth, 10 GigE. Alternatively several physical Gigabit Ethernet connections can be combined into one logical channel, e.g. using the so-called Link Aggregation Control Protocol (LACP). Currently, there are cameras with two GigE interfaces (dual GigE) available allowing the transfer bandwidth to be almost doubled.

8.4 Line-Scan Cameras

So far in this chapter we have presumed that the camera has a matrix sensor and thus a rectangular image is captured and transferred to the computer. Such cameras are also called area-scan cameras.

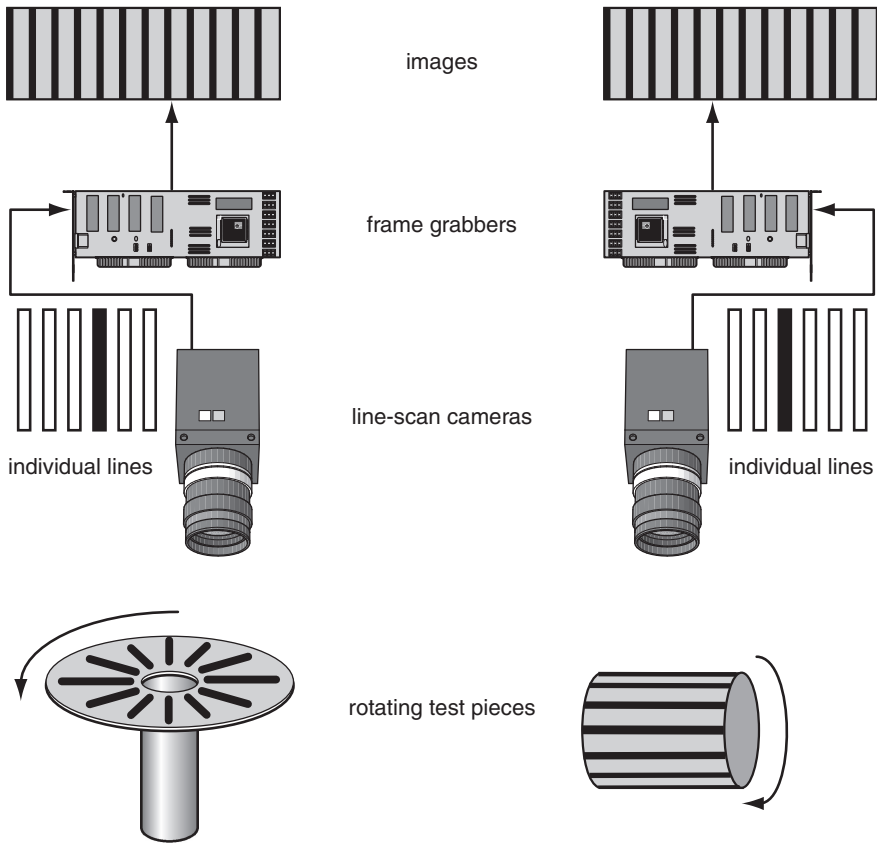


Fig. 8.15 Imaging of rotating objects by a line-scan camera

In contrast, line-scan cameras contain a line-scan sensor that has but one single line of light-sensitive elements. This sensor can have a very high resolution, typically 2,048, 4,096 or 8,192 pixels. Line-scan cameras are typically used for the inspection of moving objects or very large objects, or for monitoring continuous production processes such as wallpaper. They are ideal for the inspection of rotationally symmetrical objects, which are turned in front of the camera as shown in Fig. 8.15. Line-scan cameras often use CameraLink[®] as the image data interface.

8.4.1 Types of Line-Scan Camera Applications

Since a line-scan camera does not record a contiguous rectangular image like an area-scan camera, but only single—though rather long—lines, processing of line-scan camera images follows its own rules. Generally, there must be a relative

movement between the camera and the inspection piece otherwise the camera would always capture the same information. Depending on how this movement is translated into an image, there are different types of line-scan camera applications.

Line-wise processing: Many inexpensive line-scan camera systems process only the current line. Inspection tasks like the presence of a part, the presence of holes in a part, even simple measuring problems requiring no more than a certain sequential number of lines being bright or dark, can well be solved in this way. Such systems are often designed as *intelligent cameras*, i.e. the camera itself is equipped with a processor and memory so that the entire processing can be done within the camera. Such cameras are usually configured using special setup programs on a PC and the complete processing programs are then downloaded via serial interface or network to the camera. The advantages of this system design are the small size and the low purchasing price. On the other hand, cumbersome configuration and limited processing power restrict the complexity of possible applications. This kind of system can therefore be compared to highly developed smart light barriers.

Piece-wise processing: With this method the camera or frame grabber collects a certain number of lines and puts them together to form a rectangular image. This method is especially appropriate for applications where single objects have to be inspected whose geometry is ill-suited for area-scan cameras, e.g. large or circular objects, as in Fig. 8.15. With this kind of application, rotation and image acquisition are initiated by the same trigger signal to make sure that during the entire rotation the lines captured by the line-scan camera are collected in the frame grabber or camera. Then a rectangular digital image is obtained depicting the unrolling of the inspection piece surface of 360° or more. This image can then be evaluated in the same way as the image of an area-scan camera.

Continuous processing: The most difficult and expensive procedure can be seen as a combination of the two methods described above. As with piece-wise processing, rectangular images are used so that complex features can be calculated which require more information than a single line or simple counting of lines. On the other hand, the image is—as with line-wise processing—updated with every newly scanned line. This method makes special demands on the frame grabber, which has to be capable of appending the most recent line and discarding the oldest one in a cyclic shift register so that at any time a current rectangular image of specific size is available for processing. Processing itself has to keep pace with the scan rate, i.e. prior to the acquisition of a new line the entire current image has to be evaluated. This method is typically used to supervise and evaluate continuous material flows, for example in the steel or textile industry using specialized algorithms. The necessary transferal and computation performance make great demands on the computer hardware.

We want to discuss here the piece-wise processing which is especially interesting for PC-based systems. Even if a single rectangular image is processed, similar to applications with area-scan cameras, some peculiarities need to be observed that we will introduce in the following.

8.4.2 Spatial Resolution of Line-Scan Cameras

Images captured with area-scan cameras have a fixed spatial resolution in x and y direction for a given setup. This is determined by the captured field of view and the number of rows and columns of the sensor. With a line-scan camera, a fixed relation determined by sensor size, optics and field of view is only applicable in the direction of the sensor's line. Perpendicular to the sensor, the spatial resolution is determined by the relative movement between object and camera and the line frequency.

We already mentioned in Sect. 8.4.1 that relative movement between object and camera is necessary for capturing an image with a line-scan camera. This can be done by translation or rotation. Let's assume a work piece with 20 mm diameter is rotated in front of a line-scan camera with a rotational speed of 1/s. Let's further assume that the camera has a line frequency of 1 kHz, i.e. it captures 1,000 lines per second. During a full rotation of the part, exactly 1,000 lines are captured. With a diameter of 20 mm, a developable surface with a circumference of $\pi \times 20 \text{ mm} \approx 62.8 \text{ mm}$ is "flattened" into the digital image. Thus the spatial resolution in this direction is 0.0628 mm/pixel.

If now the rotational speed of the part is doubled, *the same surface* is captured in half the time thus only half the number of lines is created and the size of a pixel in this direction is doubled to 0.1256 mm/pixel. If, on the other hand, the line frequency of the camera is doubled to 2 kHz, the number of lines increases to 2,000 at the original rotational speed thus the pixel size is reduced by half, 0.0314 mm/pixel.

Please note that the ratio perpendicular to the motion direction along the sensor line is not changed at all. If in the original configuration because of optics and capturing distance a 50 mm stripe of the object surface is projected on the sensor line, then this length will also be projected on the sensor line with double rotational speed or double line frequency. Since this does not change the characteristics, the spatial resolution in the direction of the sensor remains the same.

Thus the spatial resolution of line-scan cameras in sensor line direction is determined just like with area-scan cameras by the field of view and the sensor resolution, perpendicular by the ratio of line frequency and relative motion.² We will discuss the difficulty of varying motion speeds in Sect. 8.4.4.

8.4.3 Illumination for Line-Scan Cameras

Besides the geometric conditions described above, line-scan and area-scan camera capturing also differ with regard to illumination. With an area-scan camera, the

² Prerequisite is that the relative motion takes place in a 90° angle to the sensor line. Usually, this makes sense.

image is often exposed for several milliseconds. This exposure time is also available for every line respectively every pixel of the image. With a line-scan camera, maximum exposure time for a line depends on the line frequency. With the rather moderate frequency of 1 kHz, each line and thus each pixel is exposed for a maximum of one millisecond (a shorter exposure time is always possible). This is far shorter than the usual time for area-scan cameras and has two important consequences:

- With a line-scan camera, a lot of light needs to be on the part which is usually quite difficult considering the geometric conditions. Since the camera only captures exactly one line, the illumination has to be precisely adjusted to get the light to where it is needed.
- Most of the time one will have to work with a wide open aperture, as we will see in Sect. 8.5.3 which reduces depth of field. It is therefore difficult to depict objects or object parts with different distances from the camera in focus at the same time.

One could say that area-scan cameras are also used to capture images in the exposure time range of 0.1–1 ms and less. But the geometric conditions make it typically easier to install high-performance illumination with area-scan applications than with line-scan applications. Most important, however, one can take short-exposure images with area-cameras using a strobe that brings a lot of light on the sensor in a short period of time. This is not possible with line-scan cameras because each line would have to be strobed individually. This would require a strobe that could operate constantly with 1–10 kHz and also the necessary light geometry. This is not possible in practice so we can state that line-scan applications often constitute a problem with regard to light efficiency.

8.4.4 Control of Line-Scan Cameras

With line-scan cameras it is often necessary to synchronize the line start externally; this is called ExSync. Because of the relations described in Sects. 8.4.2 and 8.4.3 between relative speed, line frequency and light efficiency, ExSync is necessary in many industrial applications. Depending on the type of line synchronization, the following operating modes can be distinguished:

Free-running: In this mode the camera captures lines in sequence and transfers them. The frequency with which the lines are captured is configured in the camera. This mode is comparable to the behavior of a synchronous area-scan camera.

Fixed ExSync frequency: With this method a signal source—often the frame grabber itself—sends an ExSync signal at regular intervals to trigger capture and transfer of an image line. The difference to free-running operation is that:

- The line frequency is not configured in the camera but can be controlled by the image processing system via the frame grabber. Since the image processing

system is communicating with the frame grabber anyhow for the image capturing, one can save an additional interface and a command set for configuring the camera.

- Line capture need not be continuous, but can be started and stopped by the image processing system.

Externally synchronized ExSync frequency: From the relation between relative motion, line frequency and spatial resolution described in Sect. 8.4.2, it follows that a constant spatial resolution also necessitates a constant ratio between relative motion and line frequency. The movement of an inspection piece in rotation or linear motion will rarely be completely steady. In any case, there will be a start-up ramp for each individual inspection piece rotated in front of a camera, but also afterwards rotational speed will be subject to slight variations. In the digital image these result in local changes of image geometry. To avoid this, capturing of the image line must be synchronized with the movement of the inspection piece. Usually the necessary ExSync signals are generated by a rotary encoder coupled with the motor axis. If necessary, these signals are multiplied by a frequency divider to achieve higher line frequencies than the resolution of the rotary encoder allows.

8.5 Optical Foundations

Up to now in this chapter we have discussed how the brightness information on the camera sensor is converted into a digital image to be processed by the computer. Prior to this, a suitable image has to be created on the sensor. This is the task of the optical elements, i.e. primarily that of the lens: on the CCD sensor, a sharp picture of the image scene has to be created with sufficient contrast and resolution to check the features to be inspected. The selection of the right lens depends on a number of factors, e.g. object size, distance, lighting conditions, camera model, and the characteristics of the inspection task. In this section, we will discuss first some fundamentals of optics, then properties of specific camera lenses.

We would like to emphasize that this is a simplified presentation intended to convey understanding of the basics and properties of lens imaging closely following notation and procedure common in physics, e.g. using absolute values. For the calculation and construction of camera lenses more complex processes must be used, and the inner structure of the lens system and relations between material and geometry of the lenses must be taken into consideration. For these issues we refer the interested reader to the textbooks of technical optics.

8.5.1 *f-number*

Before we come to the laws of imaging we have to introduce the concept of *f-number*, which we will need in the following.

Figure 8.16 shows the conditions behind the lens of a camera. A beam of rays falls through the aperture and illuminates the sensor. For sufficient exposure a certain amount of light per area is required. The amount of light is determined by the exposure time and the diameter of the beam of rays. The area over which this amount of light is distributed depends on the distance between lens and sensor. When an image of an infinitely (or at least sufficiently) distant object is created on the sensor, this distance is equal to the focal length of the lens, as we will see in Sect. 8.5.2.

The relation between the diameter d of the entrance pupil of a lens and its focal length f is called the *relative aperture*; its reciprocal is the *f-number* k .

$$k = \frac{f}{d}. \quad (8.2)$$

The light intensity is therefore proportional to the square of the relative aperture and inversely proportional to the square of the f-number:

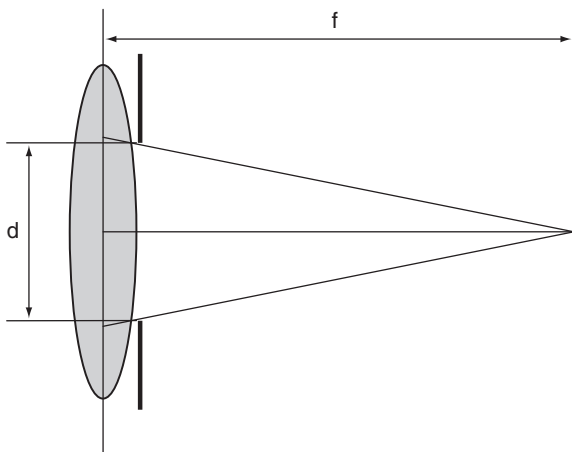
$$E \sim \left(\frac{1}{k}\right)^2. \quad (8.3)$$

This means that doubling the f-number reduces the light intensity to a quarter of its former value.

Standard aperture settings: Lenses typically have a sequence of standard aperture settings on their diaphragm ring, so-called f-stops, consisting of the following f-numbers:

1.0 1.4 2 2.8 4 5.6 8 11 16 22 32

Fig. 8.16 Relative opening of lens



Subsequent values have a ratio of $\sqrt{2}$. Since the light intensity, as we have seen above, is inversely proportional to the square of the f-number, light intensity is halved from one f-number in the series to the next. To collect the same amount of light on the sensor, the exposure time then has to be doubled.

Lens speed: The maximum relative aperture of a lens is called lens speed. It limits the amount of light which can pass through the lens and thereby the minimum illumination required to take images with this lens (besides other factors e.g. the sensitivity of the sensor).

8.5.2 Laws of Imaging

This section explains the basic laws governing the generation of an image on a sensor surface—be it a photographic film, a CCD array or the retina of the eye.

We start with the idealized case of a thin lens, i.e. a lens whose own dimensions are irrelevant for the image formation. The geometry of imaging by a thin lens is governed by the laws according to Schröder and Treiber (2002) as illustrated in Fig. 8.17:

1. Any ray that comes in parallel to the axis on one side proceeds towards a certain particular point called the focus F on the other side.
2. Any ray that arrives at the lens from the focus F' on one side comes out parallel to the axis on the other side.
3. A ray passing through the center of the lens does not change its direction.

As shown in Fig. 8.17 it follows from these three laws that all rays going out from one point A of the object meet again in one point A' behind the lens.

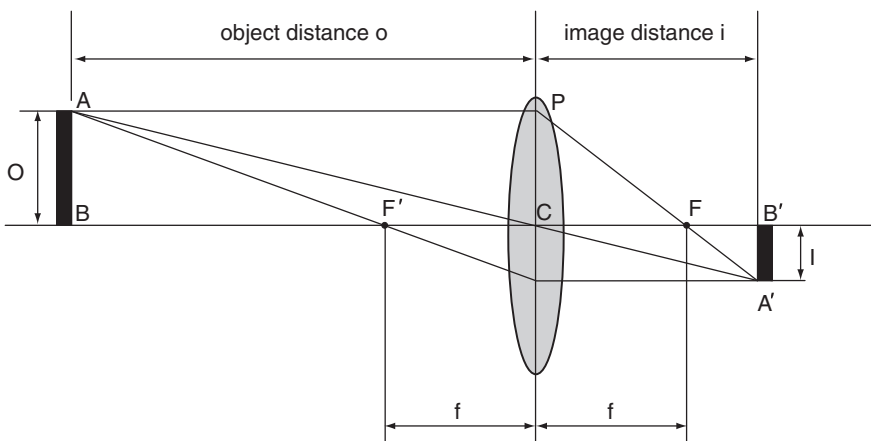


Fig. 8.17 Construction of the image of a thin lens

This means that there is a certain distance behind the midplane of the lens, called the image distance i , at which a point from the object side will again become the sharp image of a point. Closer to the lens as well as farther away this point will have a finite size, i.e. it will appear out-of-focus.

Magnification: From the similarity of the triangles ABC and $A'B'C$, it follows that the ratio of the object distance o and the image distance i has to equal that of object height O and image height I . The imaging scale, i.e. the ratio of image size to object, is then given by:

$$\frac{I}{O} = \frac{i}{o}. \quad (8.4)$$

In the same way it follows from the similarity of the triangles PCF and $A'B'F$ that with the focal length f the following relationship also holds:

$$\frac{I}{PC} = \frac{i-f}{f}. \quad (8.5)$$

Since the distance PC and the object height O are also identical, the right sides of both equations are identical. It follows that

$$\frac{i}{o} = \frac{i-f}{f}. \quad (8.6)$$

Thin lens imaging equation: From Eq. (8.6) the thin lens imaging equation can be derived by some simple arithmetic as

$$\frac{1}{f} = \frac{1}{o} + \frac{1}{i}. \quad (8.7)$$

With (8.7) the image distance can be calculated from object distance and focal length by

$$i = \frac{of}{o-f}. \quad (8.8)$$

Inserting this into the equation for the magnification (8.4) results in the following relationship for the image height depending on object height O , object distance and focal length:

$$I = \frac{f}{o-f} O. \quad (8.9)$$

Two properties of lens imaging can be deduced from this equation:

- Increasing the focal length leads to a magnification of the image. This is the reason for the length of zoom lenses.
- The image also becomes bigger when we get closer to the object. This obvious fact is illustrated by Fig. 8.18. Together with the image size, the image distance is also increased according to Eq. (8.4). This means that the distance between

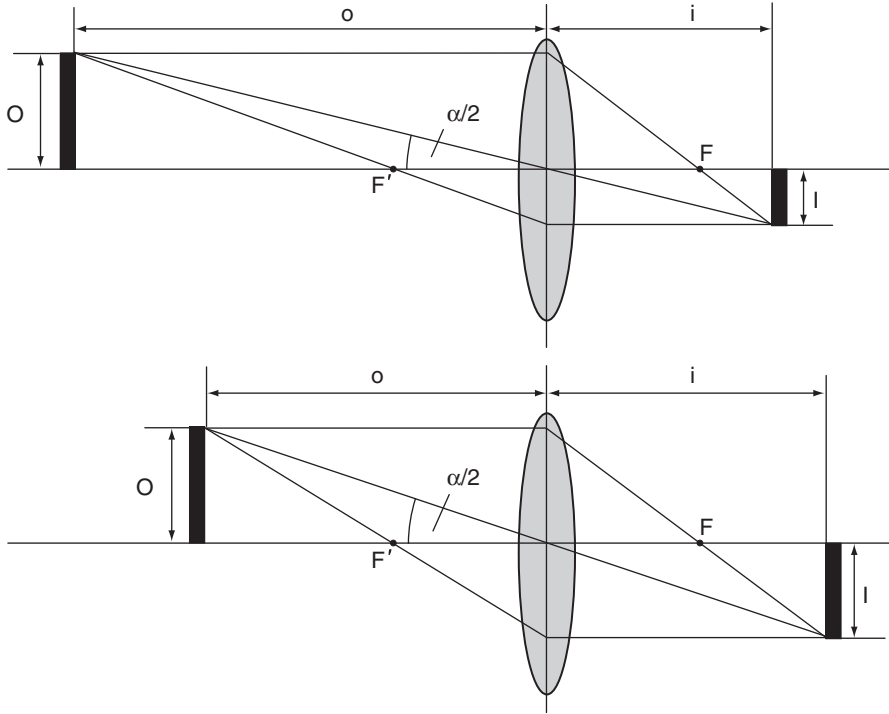


Fig. 8.18 Object distance, image distance and image size

sensor and lens has to be increased to keep the image in focus. Since the sensor is mounted in a fixed position in the camera, the lens tube is turned to move the lens farther away from the sensor.

From the thin lens imaging equation in Eq. (8.7) it follows further that an infinitely distant object will be in focus for an image distance equal to the focal length, i.e. a sharp image of an infinitely distant object will appear exactly in the focal plane:

$$\lim_{o \rightarrow \infty} \frac{1}{i} = \frac{1}{f} - \lim_{o \rightarrow \infty} \frac{1}{o} = \frac{1}{f}. \quad (8.10)$$

An interesting special case occurs when the object distance equals twice the focal length:

$$i = \frac{of}{o-f} \text{ for } o = 2f, \quad (8.11)$$

$$i = 2f. \quad (8.12)$$

This is called *symmetrical imaging* because the image has the same size as the object and is created in the same distance from the lens.

Field of view: From Eq. (8.4) for the imaging scale and Eq. (8.7) for the relationship with the focal length it can be computed how large the camera's field of view will be. Or in other words: how large an object can be fit into the maximum image size given by the size of the sensor?

From Eq. (8.4) it follows for the size O of the object with object distance o , image distance i and image size I :

$$O = \frac{o}{i} I. \quad (8.13)$$

For the image distance i it follows from the thin lens imaging equation (8.7):

$$i = \frac{of}{o - f}. \quad (8.14)$$

Which results in the following equation for the size of the object for a given image size I :

$$O = \frac{o - f}{f} I. \quad (8.15)$$

Usually the following approximation is used, which does not lead to significant errors for practical working distances:

$$O \approx \frac{o}{f} I. \quad (8.16)$$

Table 8.8 lists the image sizes for various sensor sizes when using a lens with a focal length of 16 mm and an object distance of 100 mm. It should be added that the table—like all relations discussed in this section and the following tables—refers to the ideal case of a single thin lens. The actual values for the more elaborate lens configurations of commercial camera lenses may differ from these values.

Angle of view: As can be seen in Fig. 8.18, the following holds for the angle between the optical axis and the marginal ray to the farthest point of the object O with approximation from Eq. (8.16):

$$\begin{aligned} \frac{O}{o} &= \tan\left(\frac{\alpha}{2}\right) \\ \alpha &= 2 \arctan\left(\frac{O}{o}\right) \\ \alpha &= 2 \arctan\left(\frac{I}{f}\right). \end{aligned}$$

The maximum opening angle is achieved when the object image fills the sensor completely. In this case, the image size I according to Fig. 8.17 equals half the length of a sensor side. $I = S/2$. The opening angle then is a function of sensor size and focal length:

$$\alpha = 2 \arctan \left(\frac{S}{2f} \right). \tag{8.17}$$

Table 8.9 lists the angles of view for some typical focal lengths and sensor sizes. The angles of view have been computed for the longer side of the sensors (typical sensor sizes are listed in Table 8.1). In connection with Table 8.8 we already mentioned that the actual values for commercial lenses differ from the values computed here for the ideal case of a thin lens. The values in parentheses have been taken from Pentax (2008) to show that these differences are typically quite small.

Thick lenses: To correct aberrations, real camera lenses consist of an entire system of individual lenses. The idealization of a lens being infinitesimally thin compared to its focal length cannot be maintained for such a lens system. However, the laws of lens imaging will not change fundamentally, except that one has to relate all distances to the principal planes of the lens, as shown in Fig. 8.19. For practical purposes one simply uses the relationships derived for thin lenses with the corresponding characteristic values of the actual lenses.

8.5.3 Depth of Field

The thin lens imaging equation (8.8) gives a precise relation between focal length, object distance and image distance. A specific distance between lens and sensor is equivalent to a fixed image distance from which the object distance can be calculated exactly. Only objects at this distance from the lens will appear with mathematically exact sharpness in the image. In practice though, there will be no *noticeable* blur within a certain range of object distances. We speak of a blur when a punctiform object is not depicted as a point but as a *circle of confusion*. Figure 8.20 shows the conditions within the camera for the following prerequisites:

Table 8.8 Field of view depending on sensor size

Sensor			Field of view	
Measurement	Width (mm)	Height (mm)	Width (mm)	Height (mm)
2/3"	8.8	6.6	55.00	41.25
1/2"	6.4	4.8	40.00	30.00
1/3"	4.8	3.6	30.00	22.50
1/4"	3.6	2.7	22.50	16.88

Lens format: 1" (see Sect. 8.1.6 for more information about the relationship between lens and sensor size)
Focal length: 16 mm
Object distance: 100 mm

- An object at distance o_0 from the lens imaged in focus at a distance i_0 behind the lens.
- An object closer to the lens at distance o_1 would be imaged in focus at a distance i_1 behind the lens.
- Since the marginal rays of a point on the second object intersect the image plane before they come into focus, the point is imaged as a circle of confusion with diameter C .

As long as this circle of confusion is smaller than a single sensor element, it will illuminate only a single element. Despite the divergence, the point will remain a point on the sensor, it will not spread over several pixels. Therefore, objects within a certain range of distances will appear sharp in the image. This range is called *depth of field*.

Figure 8.20 also shows that reducing the aperture will increase the depth of field. If the iris indicated behind the lens is closed further, i.e. its diameter d made smaller, the angle of divergence of the ray beam will also become smaller. In the process, the circle of confusion will shrink. If the circle of confusion was slightly larger than a single sensor element before, it can be focused to a single pixel by reducing the aperture. Good illumination therefore improves object recognition not only through improved contrast but also through improved image definition since the iris can be closed further. For the same reason, defective vision in a human will be the more pronounced the darker it is: the pupil cannot be contracted sufficiently to achieve adequate depth of field.

Whether the point of an object is in focus depends on the following factors as can be seen in Fig. 8.18:

- Aperture.
- Image distance i_0 of the object in focus.
- Image distance i_1 of the object closer to (or farther from) the lens.
- Acceptable diameter C of the circle of confusion.

The last factor, C , depends on the sensor used and is, to a certain degree, arbitrary. A typical value is half the size of a sensor element because as long as a point from the image scene is not larger, it will never be able to illuminate two

Table 8.9 Opening angle against focal distance and sensor size

f (mm)	Sensor size			
	1/4"	1/3"	1/2"	2/3"
8.5	23.91 (24.02)	31.53 (31.87)	41.26 (42.09)	54.74 (56.49)
12.5	16.39 (16.53)	21.74 (21.97)	28.72 (29.12)	38.78 (39.56)
16	12.84 (12.70)	17.06 (16.91)	22.62 (22.48)	30.75 (30.72)
25	08.24 (8.23)	10.97 (10.95)	14.59 (14.55)	19.96 (19.87)
50	04.12 (4.13)	05.50 (5.48)	07.32 (7.32)	10.06 (10.03)
75	02.75 (2.75)	03.67 (3.65)	04.89 (4.88)	06.72 (6.57)

All values in °

Values in parentheses for commercial lenses according to Pentax (2008)

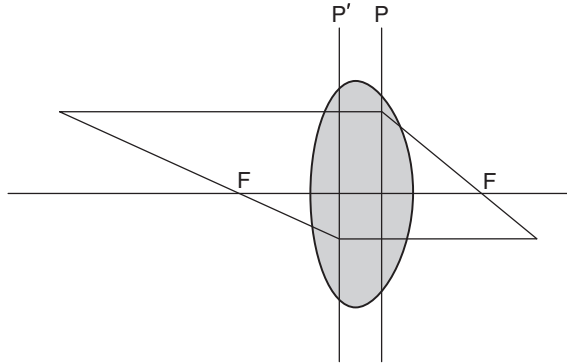


Fig. 8.19 Principal planes of a thick lens

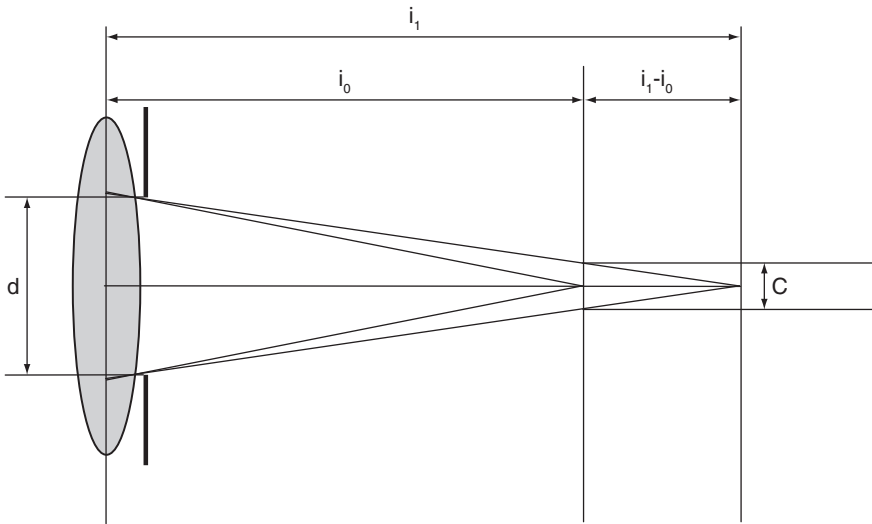


Fig. 8.20 Creation of a circle of confusion

pixels on an interline transfer sensor simultaneously, i.e. it cannot cause noticeable blur. There is some confusion in the literature about the computation of the depth of field, which is why we will derive the necessary formulas in Appendix A.2. The results needed for practical work are summarized in the following.

Computation of the depth of field: Objects outside a nominal distance o_∞ will always be in focus in the focal plane. For smaller nominal distances, the approximation given in Eq. (8.18) will not lead to significant errors.

$$\Delta o \approx \frac{2Ck\alpha_0(o_0 - f)}{f^2} \quad \text{for } o_0 \ll \infty, \quad (8.18)$$

$$o_{\infty} > \frac{f^2}{Ck}. \quad (8.19)$$

with:

o_0 Nominal object distance upon which the lens has been focused.

o_{∞} Nominal distance outside of which all objects appear in focus.

Δo Size of depth of field.

f Focal length of lens.

k F-number.

C Acceptable diameter of the circle of confusion, depends on the resolution of the sensor; for CCD arrays typically half the dimension of a sensor element.

Dependence of depth of field on f-number: It follows from Eq. (A.25) that the lower bound $o_{\min}$ becomes smaller with increasing f-number k . This means that the closest point where an object will be in focus moves closer to the lens. Inversely, $o_{\max}$ will increase with increasing f-number so that the farthest point still in focus moves away from the lens. Both effects contribute to the depth of field so that the entire depth of field will increase with increasing f-number. This can also be seen from Fig. 8.20: if the diameter of the aperture is reduced, the beam of rays contracts and so does the circle of confusion. In other words: the better the scene is illuminated, the larger the range will be inside which objects are in focus since the iris can be closed further.

Table 8.10 gives the lower and upper limit distances and the size of the depth of field for a lens with a focal length of 8 mm, a nominal object distance of 100 mm and an acceptable blur of 5 μm for all settings of the standard f-number series. The increase of the depth of field with increasing f-number can be clearly seen.

Dependence of the depth of field on the focal length: It can be easily shown that increasing the focal length increases the lower limit distance while the upper limit distance is reduced (see Appendix A.2.3). This means that the limit points

Table 8.10 Depth of field depending on f-number

f-number	$o_{\min}$ (mm)	$o_{\max}$ (mm)	Depth of field (mm)
1.00	99.29	100.72	01.44
1.40	99.00	101.02	02.01
2.00	98.58	101.46	02.88
2.80	98.03	102.05	04.03
4.00	97.00	102.96	05.75
5.60	96.13	104.19	08.06
8.00	94.56	106.10	11.54
11.00	92.00	108.59	15.91
16.00	89.69	112.99	23.31
22.00	86.35	118.78	32.44
32.00	81.30	129.87	48.57

Focal length: 8 mm

Nominal object distance: 100 mm

Acceptable blurring: 5 μm

Table 8.11 Depth of field depending on focal length

Focal length	$o_{\min}$ (mm)	$o_{\max}$ (mm)	Depth of field (mm)
2.60	46.45	∞	∞
8.00	89.69	112.99	23.31
12.00	95.34	105.14	09.80
16.00	97.44	102.70	05.25
25.00	99.05	100.97	01.92
50.00	99.84	100.16	00.32
75.00	99.96	100.04	00.07

F-number: 16
Nominal object distance: 100 mm
Acceptable blurring: 5 μm

Table 8.12 Depth of field depending on nominal distance

Nominal distance	$o_{\min}$ (mm)	$o_{\max}$ (mm)	Depth of field (mm)
100.00	96.75	103.48	6.73
200.00	186.60	215.47	28.87
500.00	421.45	614.54	193.10
1000.00	726.03	1606.07	880.04
2000.00	1136.81	8309.29	7172.48
5000.00	1721.09	∞	∞

F-number: 11
Focal length: 12 mm
Acceptable blurring: 5 μm

between which objects are in focus move towards each other so that the depth of field is reduced. This is the reason why zoom lenses have a very small depth of field while capturing far away objects. Table 8.11 lists the depths of field for the focal lengths of typical lenses used in digital image processing. Whereas the range of the 2.6 mm lens extends outward to infinity, the depth of field of the 75 mm lens has a size of only a few tenths of a millimeter.

Dependence of depth of field on distance: With increasing nominal distance o_0 both limit distances increase also, but the upper faster than the lower as can easily be seen by inserting values into the respective equations. The depth of field will therefore increase with the nominal distance. Table 8.12 shows this using a 12 mm lens. The increase of the depth of field with increasing nominal distance makes the well-known ∞ setting possible.

8.5.4 Typical Capturing Situations

Taking the distances between object and camera or lens, three typical capturing situations can be distinguished:

- Distant objects
- Macro image
- Micrograph

Distant objects: When the object- distance is much larger than the focal length of the lens, this is referred to as distant objects. In this case image distance and focal length are about equal. This is the typical situation in photography and filming, and also in many image processing applications. Frequently in this type of application rather large objects have to be inspected without special demands on measuring accuracy. Because of the relatively small focal length, the depth of field in this imaging situation is quite large making positioning less critical.

Example: For a lens with a focal length of $f = 16$ mm, an f-number of $k = 11$ and an object distance of $o_0 = 1,000$ mm, according to Eq. (8.18) the depth of field is larger than 440 mm for an acceptable blur of $5 \mu\text{m}$.

Macro image: It is characteristic for macro images that the image distance is roughly equal to the object distance. It follows from Eq. (8.12) for symmetrical imaging that both values are equal to twice the focal length. Because of the reduced depth of field with reduced object distance, it is obvious that only a very small range is in focus. This is also close to the limit where focused images are possible with normal camera lenses.

Example: If we take the previous example for distant objects and change only the object distance, we see that with a focal length of $f = 16$ mm, an f-number of $k = 16$ and an acceptable blur of $5 \mu\text{m}$, the depth of field for the symmetrical object distance $o_0 = 2f = 32$ mm has a focus range of only 0.22 mm.

Micrograph: For micrograph images the image distance is significantly greater than the focal length and thus the object distance almost equals focal length. This is the typical situation with a microscope where the depth of field is even more reduced.

The object distance is related to the image distance by the magnification according to Eq. (8.4):

$$o = \frac{I}{O} i. \quad (8.20)$$

From a magnification factor of 50 ($I = 50 O$) it follows that $o = 50 i$. Since the thin lens imaging Eq. (8.7) shows that

$$o = \frac{if}{i - f}. \quad (8.21)$$

It can be calculated that the object distance in this case has to be $1.02 * f$. With otherwise unchanged data ($f = 16$ mm, $k = 11$, $C = 5 \mu\text{m}$), we obtain a depth of field of just $2 \mu\text{m}$. If one takes into consideration that such images usually require fully opened aperture ($k = 1$) to achieve the necessary light efficiency, this is reduced to $0.2 \mu\text{m}$. Thus it is obvious that microscopes can only image in focus at a certain distance.

Such an enlargement cannot be achieved with ordinary camera lenses. The extreme image distance would not allow the image to be focused at all.

8.5.5 Aberrations

Simple lenses have a number of aberrations which can be corrected to a large extent by modern lens systems. We mention these for completeness' sake. In practice they are of no importance except in special applications.

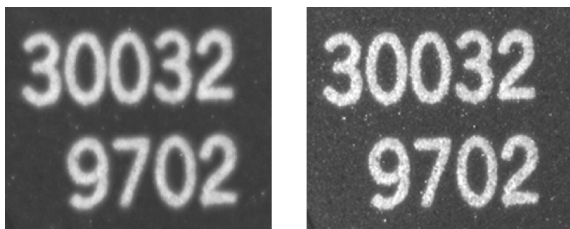
Chromatic aberration: Light of different wavelengths is refracted differently, therefore the focal length is not identical for different colors. Theoretically, a simple lens will focus only monochromatic light correctly so that color fringes appear in the image. Even corrected lens systems (achromatic or apochromatic lenses) will work without aberration only for selected wavelengths, not over the entire frequency range. For high-accuracy gauging tasks monochromatic light is therefore preferable.

This aberration is more pronounced the larger the difference in wavelengths and can cause interesting effects. As described in Sect. 8.1.6, CCD sensors are sensitive for near infrared. Figure 8.21 shows an IC with laser-etched characters; on the left, illuminated by a fluorescent ring-light alone, on the right also by infrared LEDs. Without adjusting the lens settings, sharpness was increased by adding the infrared light. The wavelength of the infrared LEDs of about 950 nm is much larger than that of the fluorescent light of approx. 600 nm. Obviously the lens focuses the wavelength of the LED better at the selected setting. In general, the reverse effect is more common. Since lenses are usually designed and adjusted for imaging under visible light, uncontrolled infrared radiation will usually reduce sharpness. The naked eye will not see any difference since we cannot perceive the infrared light.

Spherical aberration: The angle between the surface of the lens and parallel rays far away from the optical axis is more oblique than for rays close to the optical axis. Therefore, the outer rays will be refracted more strongly. This aberration can be reduced by increasing the f-number since the smaller aperture will admit only rays close to the optical axis.

Curvature of field: Since the image is actually created on a curved plane, the image margins appear blurred.

Fig. 8.21 Changing image sharpness by infrared illumination



Distortion: Imaging of a rectangular grid will result in straight lines being curved outward or inward; this increases with their distance from the optical axis. This effect is caused by different magnification scales for different imaging angles. Typical distortion values are between 0.2 and 1.0 %; this means that deviations up to five image pixels can occur.

8.5.6 Lens Determination

We have said a lot about the laws of imaging and their equations so far in this section because we think it is important to be familiar with these theoretical foundations in order to understand the properties of optical systems. In practice, however, there are a number of terms and concepts we have not yet mentioned. Manufacturer manuals and catalogs explain in more detail than we can how to handle lenses; however, we will give a brief overview.

Flange back length (mount): The flange distance or flange back length, probably better known as mount, is the distance from the back of the lens flange to the sensor plane. The mount is given for cameras and lenses. Usually, C-mount cameras and C-mount lenses are used nowadays with the C-mount flange back length of 17.526 mm. The CS-mount flange back length is also standardized at 12.526 mm; however, there are hardly any CS-mount cameras or CS-mount lenses today.

From the point of view of the camera, the flange back length is the *smallest possible* distance between lens and sensor because the back of the lens rests on the camera casing at this distance. This is a structural property of the camera that cannot be changed.

From the point of view of the lens, however, the flange back length is the *greatest permissible* distance away from the sensor at which it can still focus on an infinitely distant object. We know from Sect. 8.5.2 that a lens system has its shortest image distance when the object to be focused on is infinitely far away. For any smaller object distance, the lens has to be farther away from the sensor. The shortest possible distance between lens and sensor is given by the flange back length.

For a C-mount lens the sensor has to be at a distance of 17.526 mm for an infinitely removed object to be in focus. Accordingly, the sensor has to be at a distance of 12.526 mm for a CS-mount lens. It follows that a CS mount lens cannot be used on a C-mount camera because the constructional minimum distance between lens flange and sensor is 17.526 mm for a C-mount camera.

On the other hand, CS-mount cameras allow for the use of CS-mount and C-mount lenses, provided that a 5 mm extension ring or macro-focus ring is inserted between camera and lens to increase the distance between lens and sensor to the 17.526 mm required by the lens. Figure 8.22 illustrates these relations.

Close-up imaging: Figure 8.18 shows that the image distance is increased when we close in on an object. The object is imaged in focus at a larger distance

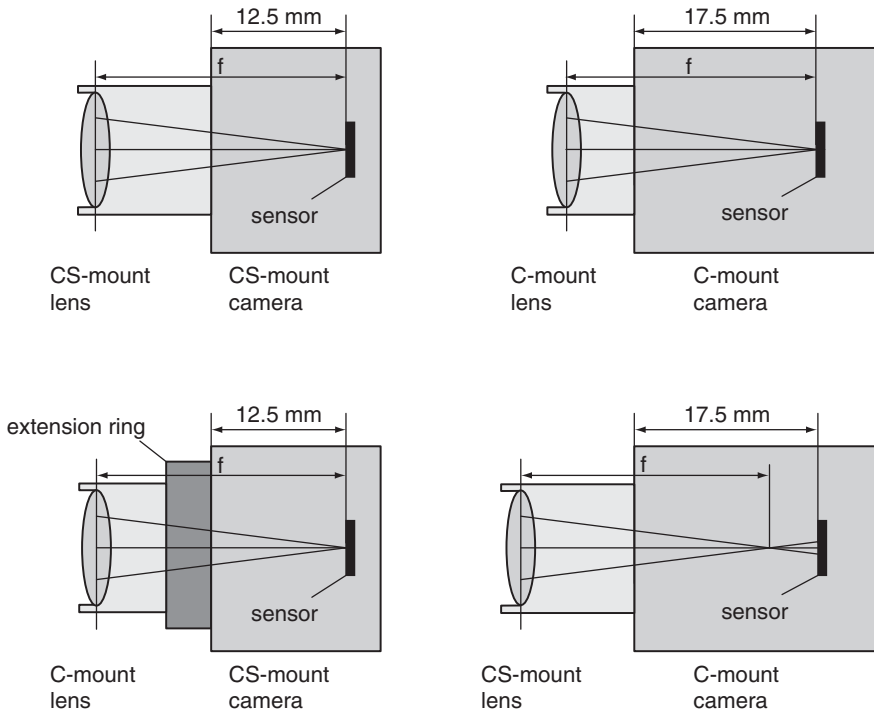


Fig. 8.22 Combination of C-mount and CS-mount flange back length

behind the lens. Therefore, the distance between lens and sensor has to be increased for close-up images. By turning the focus ring on the lens, the lens is moved away from the sensor in the lens tube. This motion is of course restricted to a certain range limiting the minimum object distance the lens can be focused on.³ To get closer to the object, the distance between lens and sensor has to be increased. This is the purpose of extension tubes or macro-focus rings which are available in various thicknesses. Typical sizes are, for example, one, two, five, ten millimeters etc.

When using extension tubes, one has to be aware that the depth of field is reduced. Distant objects can no longer be brought into focus. The minimum distance between lens and sensor is typically equal to the focal length of the lens so that infinitely distant objects can be sharply imaged. Using a 5 mm extension tube with a 16 mm lens increases this distance to 21 mm. Smaller image distances are no longer possible. It follows from the thin lens imaging Eq. (8.7) for the object distance o with image distance i and focal length f :

³ There is another obvious limitation: objects inside the focal length of the lens cannot be sharply imaged at all because their marginal rays diverge behind the lens. The image distance in Eq. (8.7) would become negative.

$$\frac{of}{o-f} = i \geq i^*, \quad (8.22)$$

$$o \leq \frac{fi^*}{i^* - f}. \quad (8.23)$$

and with $f = 16$ mm and $i^* = 21$ mm follows $o = 67.2$ mm. Hence, the lens can no longer be focused on objects farther away.

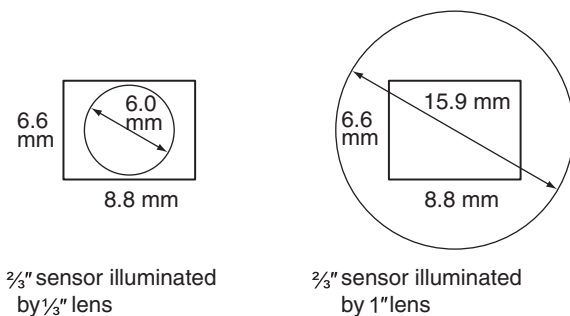
Relationship between lens size and optical format: We have already mentioned in Sect. 8.1.6 that there are CCD sensors of different size and that the sizes of sensor and lens are related and have to be adapted to each other. To do this, a size category is used rather than the absolute size. A size of one inch for a lens means it illuminates a light circle that is large enough for a one-inch sensor. Thus a $1/3''$ -lens illuminates a light circle large enough for a $1/3''$ -sensor. It follows that lenses can be used together with sensors of smaller size but not vice versa since the lens would not illuminate the sensor completely. Figure 8.23 illustrates this relationship.

8.5.7 Special Lens Types

In this last section on optical elements we will introduce some special types of optical devices whose specific properties make them particularly well-suited for certain applications.

Telecentric lens: This type of lens is mainly used in gauging applications. It has the important property that within a certain range, the telecentric range, different distances between object and lens do not lead to changes in the size of the image. This is possible only if the path of the rays on the object side of the lens is parallel. Under this condition, all rays of the image will pass through the focus. This telecentric ray path can be achieved by placing the exit pupil in the focal plane of the sensor-side of the lens. Note that the object to be imaged with a telecentric lens cannot be larger than the lens itself. Otherwise the marginal rays in Fig. 8.17 cannot enter the lens parallel to each other, again changing their angle depending on the distance. Therefore, telecentric lenses can only be built for

Fig. 8.23 Relationship between lens size and sensor size



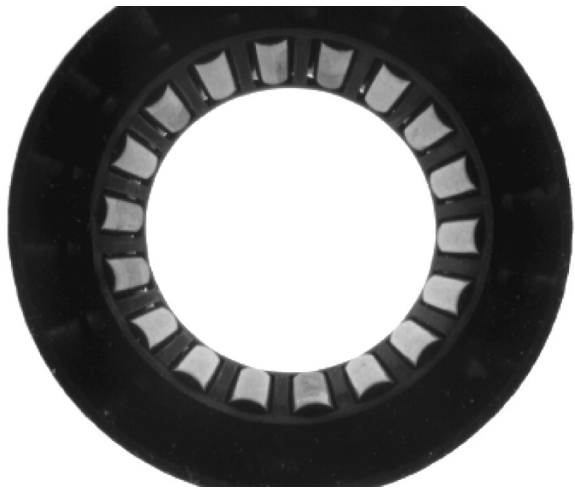
relatively small objects. For an object-size of more than 100 mm, these lenses achieve a length of more than half a meter and a weight of about four kilograms.

Fisheye lenses: Fisheye lenses are ultra-wide-angle lenses. They are constructed for an angle of view of more than 90° . This allows for 360° images of objects. Images taken with a fisheye lens show an extreme barrel distortion, making precise measurements impossible. They are eminently suitable for presence verification inside rotationally symmetrical objects. Figure 8.24 demonstrates this with an image of a roller bearing.

Endoscopes: Most people will probably associate the term “endoscopy” with medical instruments. Endoscopes are used in medicine to examine body cavities. Similarly, endoscopes can be used in technical applications to inspect inaccessible areas of a part. Pschyrembel (2007) describes an endoscope as a tubular instrument with an optical system consisting of objective lens and ocular, or eyepiece, and an illumination device usually based on a cold light source whose light is reflected into the endoscope. Most common are glass fiber endoscopes. Light is conducted along the glass fiber bundles by multiple total reflection, first from the light source to the object, then back to the sensor.

Special optical devices: A variety of custom optical components has been designed for specific purposes. Some of these are available ready-made by catalog, frequently though they are specially designed and custom-made. Therefore, it is impossible to even try and give an overview here and we restrict ourselves to some hints for the manifold possibilities of such optical devices. One example is provided by two-point measuring optics, which reflect images from two points of an object lying far apart into a single lens of smaller diameter. The resolution effective for measuring can thus be considerably increased since the entire area of the inspection piece between the two measuring points does not appear in the image. Other examples are prism heads for inspecting bore holes or conical

Fig. 8.24 Image of a roller bearing with a fisheye lens



mirrors for imaging the outside surface of cylindrical objects with a single area-scan camera.

8.6 Illumination Technology

Finding an illumination setup for a concrete application will always require experimentation with the actual test pieces. It is therefore very difficult to give general guidelines for this important design process. Successful experimentation, however, is made much easier by fundamental knowledge about different lighting methods and devices. That is why in the application examples in previous chapters we have usually remarked on the illumination techniques used. These remarks will now be supplemented by an overview of typical lighting methods.

8.6.1 *Light Sources*

Considering the variety of lighting configurations, it is somewhat surprising that only a small number of different light sources are commonly used. Normal incandescent lamps are practically not used because of the heat developed and the marked variation of their brightness with the power supply frequency. This leaves the following important light sources:

- Light-emitting diodes
- Lasers, usually in the form of laser diodes
- Fluorescent tubes
- Halogen lamps

In the following we will briefly describe the properties and typical applications of these light sources.

Light-emitting diodes: Nowadays, Light Emitting Diodes (LED) are the most common light sources in industrial image processing. Lights employing LEDs usually consist of many LEDs that can be arranged at will and allow for a large number of geometric variations. They can be arranged in rows for line or beam lights, in a rectangular array as light fields, as ring lights of almost any size, and be used as light sources in dome illumination. For point illumination, LED spots can be used; furthermore, LED light can also be injected into glass fiber-optic light guides. Unlike light bulbs or fluorescents, LEDs emit their light in a rather narrow light cone. Therefore, with directed illumination one has to keep in mind the alignment of the LEDs.

The possibility of controlling every single LED individually allows for switched illumination setups which can, for example, light an object from various

angles in a predefined sequence. By comparing images illuminated with different angles of incidence, surface defects can be detected.

LEDs send out light in narrow wavelength bands in red, green, blue, ultraviolet or infrared. The narrow wavelength bands make it possible to shield ambient light of a different wavelength with optical filters, thus enhancing the image contrast. Although LEDs cannot produce white light per se, white-light LEDs are very common nowadays. There are two ways to create white light with LEDs. One is to use blue LEDs with a fluorescence layer that converts part of the blue light into a broad fluorescence spectrum. The converted light is yellow and combined with the rest of the blue light we obtain white light. This kind of LED frequently has a slightly bluish, cold white. The other kind is a combination of a red, green and blue LED each. With additive color mixture the light of the three LEDs is a white light.

A big advantage of LEDs is their long life span; however, an LED loses some of its luminosity before breaking down. LEDs have low power consumption since they produce little heat compared to other light sources. However, heat is the greatest threat for an LED's life span; therefore, heat dissipation is important. Finally, LEDs are very robust when it comes to shocks and vibrations. Maybe the greatest advantage of LEDs is that they can be flashed almost wear-free. Since LEDs quickly react to a change in voltage, they are suited like no other light source for strobe illumination.

Laser: Laser light is especially well suited for structured lighting because of its excellent focusing. Typical applications are laser line illuminations for checking part geometries without gray level contrast. An example of such an application is described in Sect. 9.4. The presence of dark holding pins on equally dark plastic is checked using the characteristic reflections created by a laser line.

Fluorescent tubes: Today, fluorescent tubes are mostly used for large-area lighting to achieve homogeneous illumination over larger distances. In image processing, however, it has to be noted that common fluorescent tubes show periodical brightness variations with the power supply frequency which can have very unpleasant effects in the camera image. Therefore, one should always use high-frequency fluorescents, which have the welcome added effect of enabling flicker-free starting. This is very important when fluorescent tubes are used in switched illumination systems.

Fluorescent tubes can be used in very different geometrical arrangements. For example, area lights can be constructed by mounting several tubes under translucent plastic. From the outside, the plastic appears as a uniformly lighted plane. This kind of area light can be used for diffuse front lighting as well as for back lighting for gauging tasks. Another possible geometry is represented by ring-lights, usually with a diameter of several centimeters. A drawback of this type of ring-light is that they are not absolutely uniform because the starter has to be mounted somewhere along the ring.

Halogen lamps: Because of their “uncomfortably” white light, they are also called *cold light sources* although they are anything but cold. They are often used in connection with fiber optic light guides. A fiber-optic light guide with a spot head can be used to direct light to specific areas of an object; ring-shaped light

guides often have built-in polarization filters as described in Sect. 5.2.1. Rectangular light guides similar to the area lights mentioned above distribute the light from the fiber to a large area and serve to build very uniform white back lighting.

Since halogen lamps use DC power supply, the problem of brightness variations caused by power line frequency is not an issue. However, aging of the lamps requires regular recalibration or use of regulated light sources.

Quite similar to halogen lamps in structure are xenon lamps that are used for strobe lighting because brightness and frequency of these lamps can be well controlled. The brightness of xenon strobe lights can surpass the brightness of LEDs.

8.6.2 Front Lighting

There are two basic illumination setups, front lighting and back lighting. In a front lighting setup, the illumination is on the same side of the object as the camera. Depending on shape and position of the light source and the angle of incidence, very different effects can be achieved using front lighting.

Information in this section and the following has in part been taken from Illuminating Solutions (1996) (courtesy of Dolan-Jenner Europe B.V.).

Diffuse front lighting: Diffuse lighting is lighting without a detectable direction. Thus shadows and strong reflections are avoided. Diffuse lighting is created when light is radiated from a large area. This can be achieved by placing the light source behind opaque glass or plastic, or by reflection from a white screen. The principle is shown in Fig. 8.25.

Directional front lighting: With this kind of front lighting the light source is placed in such a way that the light does not hit the surface in every direction but under a certain angle. There are two alternatives for placing the camera:

Bright field illumination: In bright field illumination the camera is positioned in the (theoretical) angle of reflection of the incident light. If the illuminated surface has the reflection properties of a mirror, the camera is hit fully by the reflected

Fig. 8.25 Principle of diffuse front lighting

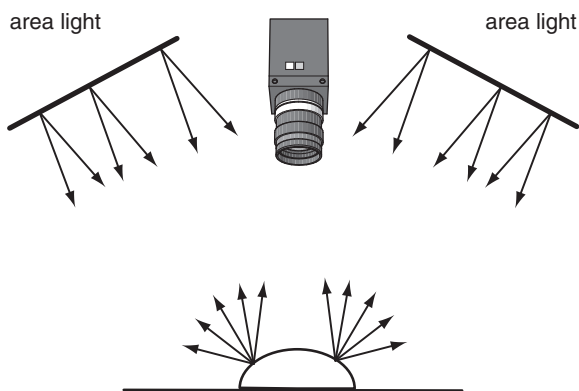
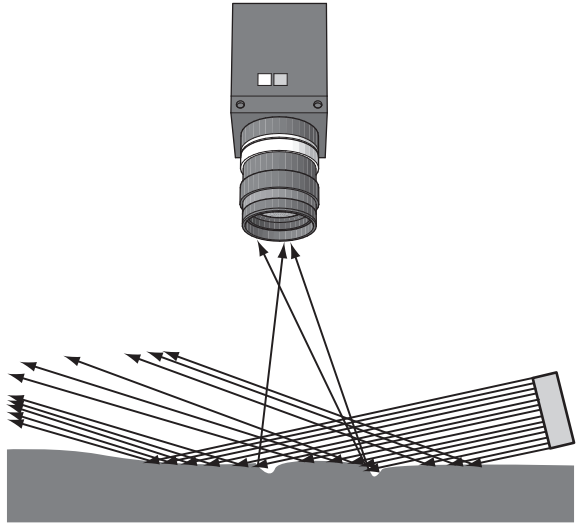


Fig. 8.26 Effect of dark field illumination on disturbances of the surface



light so the surface shows maximum brightness, hence the name *bright field*. Bright field illumination was used in Sect. 9.5 for inspecting bore holes. Of course, the bore holes do not reflect any light so they contrast very well with the polished metal surface.

Dark field illumination: In dark field illumination the camera is positioned outside the direct direction of reflection of the incident light. The name derives from the fact that an entirely smooth surface will appear dark since the light will be reflected completely past the camera. However, disturbances in the surface appear bright since the angle of reflection is changed.

Dark field illumination was used in Sect. 5.3 to make the indentations visible that are created when numbers are stamped into metal. Because of the diffuse reflection on the surface structure of the metal, the surface appears evenly bright. Marked indentations or elevations lead to different reflection angles and shadows creating a marked contrast to the surface, as can be seen in Fig. 8.26.

In extreme cases, the angle of incidence can be made so low as to be almost parallel to the surface, for example by using a line light guide to guarantee straight-lined incident light. This will create maximum contrast on textured surfaces and sharply defined shadows. This method was used in Sect. 7.4 to gauge the teeth of a rubber tooth belt. Humans usually turn and tilt an object, viewing it from different angles to discern surface structures. A similar effect can be achieved with much reduced mechanical effort by mounting various light sources in different heights and switching them in sequence. Such dynamic illuminations are most easily constructed using LEDs, for example in the form of a lighting dome which allows for diffuse illumination from all sides as well as for radial and concentric lighting arrangements.

Polarized light: Polarizing filters can be used to avoid reflections. Polarized light oscillates in only one direction perpendicular to its propagation direction. If a

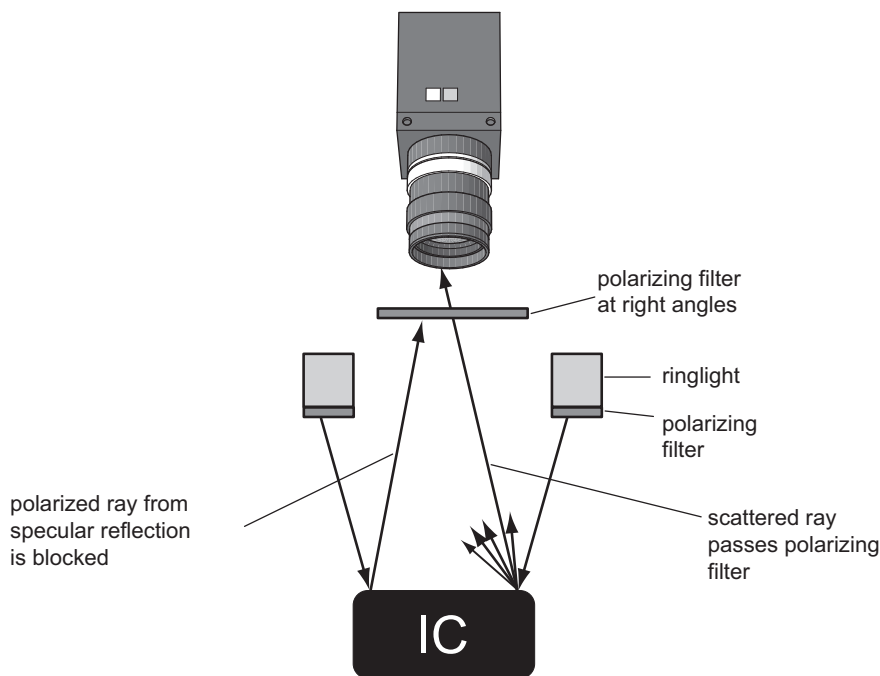


Fig. 8.27 Ring light with cross-polarizing filters

surface is illuminated using polarized light, all rays reflected through specular reflection will be completely polarized. A polarization filter at right angles to this polarization direction will block these rays. The reflections will then not be visible in the camera image.

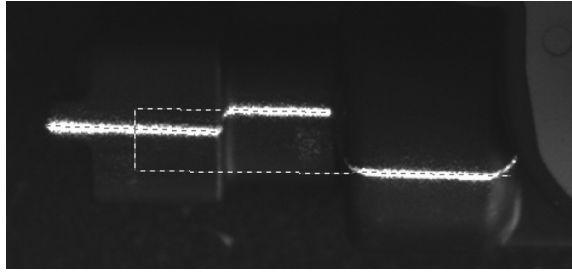
Ring lights: Ring lights provide intense shadow-free light along the optical axis of the camera. They are often combined with polarizing filters to remove problematic reflections. This illumination method was used for the recognition of laser-etched characters on an IC in Sect. 5.2.1. Figure 8.27 shows this illumination setup.

Ring lights are built in very different forms. There are ring-shaped fluorescent tubes, ring-shaped light guides for fiber optics connected to cold light sources and ring lights assembled from LEDs. The latter method was used to build a large ring light for the flange inspection in Sect. 9.3.

Axial/in-line illumination: Using special attachments or adapters it is possible to guide light into the ray path of a lens. This allows, for example, illuminating the inside of bore holes. Fiber optic endoscopes also apply this method to guide light to the object. The light is injected into the fiber bundle and illuminates the inaccessible object through the lens. The light reflected from the object is then guided back to the sensor.

Structured lighting: Structured lighting denotes the projection of lines, grids or circles onto an object. From the reflection, three-dimensional characteristics of the object can be determined. It also allows for gauging and presence verification

Fig. 8.28 Height gauging with structured lighting



without apparent brightness contrasts. Besides fiber optic line lights, laser lines are often used because of their good focusing characteristics. In Sect. 9.4 this method is used for contrast-free presence verification. Figure 8.28 shows a method for contrast-free height gauging on a black plastic part. The distance between the upper and lower light line can be used as a measure of the height of the step between the two sections of the part. The measuring lines were dashed to be discernible in the printed gray level image.

8.6.3 Back Lighting

In a back lighting setup the object is placed between light source and camera. The camera sees the object as a shadow contour. This type of lighting provides excellent contrast and is used mainly for gauging tasks. As in the preceding section, some information has been taken from Illuminating Solutions (1996) (courtesy of Dolan-Jenner Europe B.V.).

Diffuse back lighting: Diffuse back lighting is achieved in a way similar to diffuse front lighting by using a large, uniform light-emitting area, created, for example, by covering the light source with a semitransparent plate of glass or plastic. Usually LED arrays are used as light sources. Diffuse back lighting was used in Sect. 7.3 to light punched electronic components. Figure 8.29 illustrates the principle. The object blocks the rays coming from the backlight and appears in the camera image as a shadow silhouette.

Directional back lighting: A problem with diffuse back lighting is the occurrence of lateral illumination. Figure 8.30 shows this effect on the thread of a screw. In the left image, the thread of the screw is partly illuminated from the side, making stable localization of the thread edges much more difficult. In the right image, lateral illumination was shielded, thereby improving contour definition greatly.

For high-precision gauging results can be greatly improved by using telecentric back lighting. In analogy to telecentric lenses this type of illumination provides a parallel ray path produced by a collimator lens between light source and backlight. The principle is illustrated in Fig. 8.31. This type of illumination was used in the thread flange gauging application in Sect. 7.6 being the only way to make the gauging result independent of the exact distance between light source and

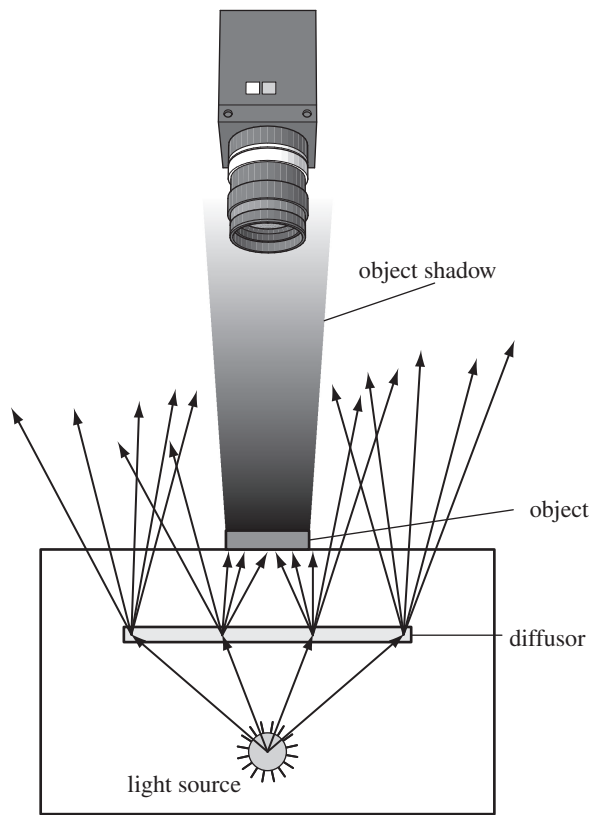


Fig. 8.29 Principle of diffuse back lighting

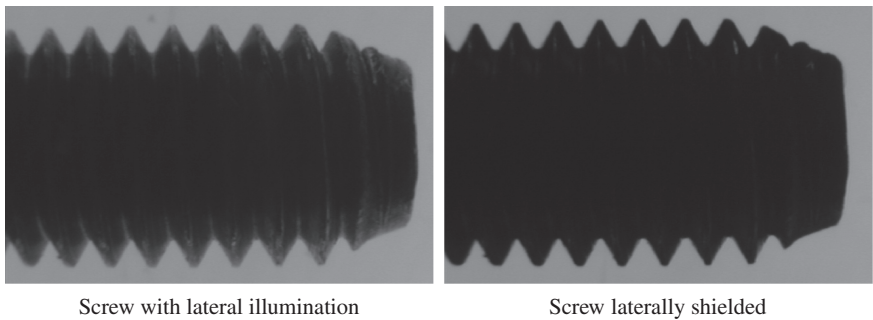


Fig. 8.30 Effect of lateral illumination on object contour

measuring plane. This is extremely important for objects that cannot be placed directly on a backlight because of their shape and size. This is the case, for

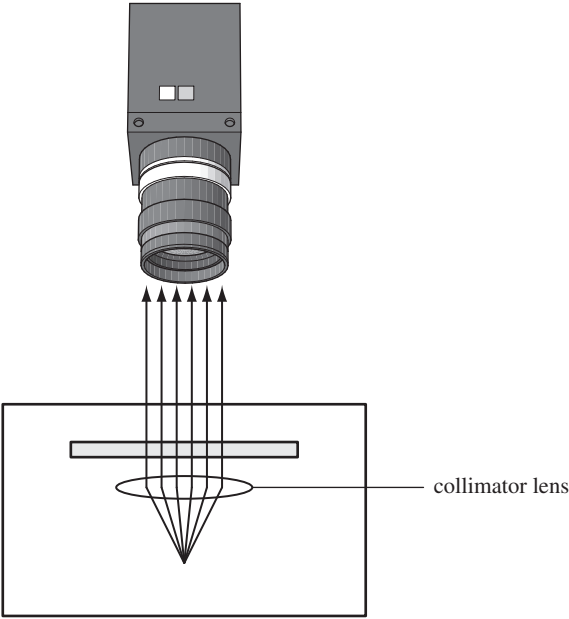


Fig. 8.31 Telecentric back lighting



Back-lit sieve Back-lit sieve with cross-polarization

Fig. 8.32 Use of polarized back lighting for inspection of transparent objects

example, with cylindrical objects like screws where the distance between the visible contour and the backlight is at least equal to the radius of the object.

Polarized back lighting: Polarizing filters can be used for back lighting as well as for front lighting. This can improve contrast on transparent pieces and even render surface tensions visible. One polarizing filter is positioned in front of the backlight, another in front of the camera. Depending on the relative angles of

the two polarizing filters, different contrast levels are produced. Figure 8.32 shows the effect on a transparent plastic sieve.

8.7 Summary

Solid state sensors: The most common type of camera used in digital image processing today employ CCD solid state sensors. The incident light sets free charge carriers in the light-sensitive part of the sensor that are collected during exposure time and then read out as proportional voltage.

Pixel: Denotes an individual sensor element on a CCD array. Pixels of typical CCD sensors have a size of less than 10 μm .

CCD sensor: Today, the still dominating solid state sensor in industrial image processing is the CCD sensor. CCD sensors often only have one charge amplifier by which the collected charges of all pixels are converted into voltage. Noise is usually lower than in CMOS sensors.

CMOS sensor: With CMOS sensors each pixel has its own charge amplifier that converts the charges integrated during exposure time into voltage. CMOS sensors are very common in the consumer market since they can be manufactured cost-efficiently and are highly integrated. This very common type of CMOS sensor has only restricted usefulness in industrial image processing because of the rolling shutter. CMOS sensors with global shutter exhibit good quality, presently; however, they are not as reasonably priced as those with rolling shutter.

Color sensors: Two types of color sensors are widely used for capturing color images:

- Three-chip color cameras with a beam splitter that guides the light according to color onto one of three solid-state sensors.
- Single-chip color cameras with a color filter array through which only the light of a single color reaches each pixel of the single solid-state sensor.

Quantum efficiency: The quantum efficiency describes how well the incident photons produce electrons. It gives the ratio between the produced electrons and incident photons.

Spectral sensitivity: The quantum efficiency of solid-state sensors depends on the wavelength. In the long-wave range the sensitivity reaches into the infrared.

Digital cameras: Cameras of this type convert sensor signals into digital values and transfer these via standardized, digital interfaces. Thus a lossless transfer of image data from the sensor to the image processing computer is ensured.

Demosaicing: For single-chip color cameras, the color value of a pixel must be interpolated from the brightness values of the neighboring pixels. This is called demosaicing and can be done in various ways.

CameraLink: This is the first digital data interface developed especially for image processing. Because of the great bandwidth, it is the interface of choice for

high-speed applications. To transfer image data to the image processing computer the latter must have a frame grabber.

FireWire: The serial data bus standardized as IEEE1394 is a high-performance and highly reliable data interface. It allows for data rates of 800 Mbit/s (IEEE1394b). It is possible to use the bus to supply connected cameras with power.

USB: The Universal Serial Bus is the single most important interface for linking personal computers with peripheral equipment. Whether the new USB 3.0 standard can establish itself in industrial image processing, remains to be seen.

GigE Vision: This standard for transferring image data uses Gigabit Ethernet as transfer medium. Gigabit Ethernet is the only digital interface allowing cable lengths of up to 100 m.

Line-scan cameras: This type of camera uses a single line of sensor elements instead of a two-dimensional matrix array. This line can be quite long, though, which allows for very high image resolutions. Line-scan cameras are frequently used for surface inspections on continuous materials or for scanning difficult part geometries. CameraLink is often used for image data transfer.

F-number: Aperture of a lens, relative to its focal length. The light intensity on the sensor is inversely proportional to the square of the f-number.

Lens speed: Maximum relative opening of lens.

Magnification: The ratio of image size I and object size O is identical to that of image distance i and object distance o , i.e. the distances between lens and object on the one hand, lens and sensor on the other hand:

$$\frac{I}{O} = \frac{i}{o}.$$

Imaging equation: The imaging equation for thin lenses gives the following relationship between focal length f , object distance o and image distance i :

$$\frac{1}{f} = \frac{1}{o} + \frac{1}{i}.$$

Field of view: The field of view of a camera is approximated with object distance o , focal length f and sensor size S by:

$$O \approx \frac{o}{f} S.$$

Depth of field: Range within which the object distance can vary without exceeding the acceptable blur. Usually half the size of a sensor element is used as the acceptable blur. The depth of field depends on:

- f-number: the larger the f-number, the larger the depth of field.
- Focal length: the larger the focal length, the smaller the depth of field.
- Distance: the larger the distance, the larger the depth of field; behind a certain limit distance all objects are imaged in focus.

Flange back length (mount): Distance between lens flange and camera sensor, thus also the minimum distance between lens and sensor. C-mount (17.5 mm) and CS-mount (12.5 mm) are distinguished. C-mount lenses can be used with CS-mount cameras by inserting an extension tube. The reverse is not possible because a CS-mount lens cannot be brought close enough to the sensor of a C-mount camera.

Extension tube: Inserting extension tubes, also called macro-focus rings, allows for bringing the lens closer to the object.

Lens size and optical format: CCD sensors are manufactured in various sizes categorized according to the diameter of the required circle of light. Accordingly, one speaks of 1"-, 2/3"-, 1/2"-, 1/3"-, 1/4"-cameras. Lenses from a higher size format can be used on smaller format cameras because they will completely illuminate the sensor. The inverse combination is not possible.

Telecentric lens: This type of lens has a parallel ray path on the side of the object. Within the telecentric range the image size is independent from distance. They are used for gauging applications. The lens has to be at least as large as the object to be imaged.

Endoscopes: Endoscopes are used to inspect inaccessible object areas such as cavities. They are usually made up of fiber optic light guides with optics and a fixture to inject light from a cold light source.

LEDs: Light emitting diodes allow for very flexible geometric illumination arrangements. They are almost free of wear and can be used to build high-intensity strobe lighting.

Laser: Because of the good focusing, lasers are well suited for structured lighting.

Fluorescent tubes: This light source is especially suitable for diffuse front and back lighting. It should always be used with high frequency starters to avoid interference with brightness variations due to the power line frequency.

Cold light sources: Halogen lamps in connection with stabilized DC power supplies provide very high quality illumination for gauging applications.

Front lighting: In a front lighting setup, the illumination is on the same side of the object as the camera. Possible setups are:

- Diffuse front lighting.
- Directional front lighting (dark field illumination) for visualizing surface irregularities.
- Ring lights, providing intense, shadow-free light close to the optical axis.
- On-axis illumination, especially for bore holes and the like.
- Structured lighting, for imaging geometrical properties without brightness contrasts.

Back lighting/transmitted light: With back lighting the object is between the light source and camera. The camera sees a silhouette of the object. Possible setups are:

- Diffuse back lighting.
- Directional back lighting.
- Telecentric back lighting for high-precision gauging applications.

Polarized light: Polarizing filters can be used to suppress specular reflections (especially in connection with ring lights) and to make structures in transparent objects visible.

References

- Illuminating Solutions (1996) Dolan Jenner Industries, Lawrence
Pentax (2008) Pentax CCTV catalogue. Pentax Imaging System Division, Tokyo
Pschyrembel W (2007) Pschyrembel Klinisches Wörterbuch, 261st edn. de Gruyter, Berlin
Schröder G, Treiber H (2002) Technische Optik, 9th edn. Vogel, Würzburg
Theuwissen AJP (1996) Solid-state imaging with charge-coupled devices. Kluwer Academic Publishers, Dordrecht

Chapter 9

Presence Verification

Viewed from the perspective of the problem to be solved, presence verification appears easily grasped and structurally rather simple. This is the reason why presence verification is often used as an introductory example to image processing. In reality, the concept of presence verification is quite difficult to define precisely: is it simply counting objects or are properties of these objects of importance? Are these properties simple features or is the entire appearance of the object to be considered? Do properties relate to individual objects only or do we have to take relationships between objects into account, for example, when doing assembly verification.

The methods used in this area are as varied as the questions above. Often methods from other application areas, like classification and gauging, are used as tools for presence verification. Another reason why presence verification was not chosen as the starting point of our introduction to image processing was that this field is characterized less by particular methods and algorithms than by the way questions are asked. If gauging is used as a tool for assembly verification, it is typically less important to verify that specific dimensions are within narrow tolerances than to check certain relationships between objects. Classification in the framework of presence verification is less concerned with *what* objects are present—which is the ultimate goal of character recognition—than whether the *right* object is there.

Because of the variety of questions asked and methods used, presence verification problems are typically rather difficult to approach. Solutions to gauging and classification tasks are usually rather similar to each other and also similar to the way a human would intuitively describe the task. The basic approach to presence verification applications is often more difficult to find and less intuitive.

In this chapter's examples we want to give an impression of the diversity of the field of presence verification and also of the fundamental way of thinking in this type of inspection problem.

9.1 Presence Verification Using PTZ Cameras

The first example for presence verification is a rather simple one since only the presence of objects and their correct fitting is checked. The inspection object is a car engine where the correct fitting of a number of sensors, clamps and clips needs to be checked. All in all, about 60 extremely different inspection features need to be checked. In many cases the inspection features cannot be separated from the background by thresholding, or can hardly be distinguished from reflections using the binary contour. Therefore, all inspection features are searched for using template matching and fitting is checked using the matching results. We selected this example less because of the relatively simple inspection strategy but to show how to capture images of so many independent inspection features efficiently.

9.1.1 *Inspection Part Geometry*

The most obvious difficulty of this inspection task is the size of the engine over which the rather small sensors, clamps and clips are distributed. When capturing the entire engine, the sensors measure merely a few pixels. When objects are this small, one can only check whether a bright spot is present or not. The differences between a correct and a faulty fitting measure less than one pixel and thus cannot be checked. It is also not possible to determine whether the bright spot is indeed the sensor in question or a light reflection. The conclusion could be drawn that individual cameras are needed for inspecting individual sensors and components resulting in a large number of hardware items.

Indeed it is possible to use considerably fewer cameras if special ones are used. Applications like this one call for pan/tilt/zoom (PTZ) cameras. They consist of a camera head mounted on a stand in such a way that it can be turned both horizontally and vertically by built-in motors. Thus the line of sight of the CCD camera integrated into the camera head can be changed while the size of the captured field of view can be manipulated simultaneously using the integrated zoom lens. These cameras are usually used for video conferences and surveillance tasks but can also be used in machine vision. This example presents a typical application for this type of camera where a sequence of detail views of the engine are captured in succession. The complex shape of the engine and the large number of inspection features necessitate a total of five PTZ cameras capturing the engine from different angles and heights.

As useful as PTZ cameras are, their use causes several difficulties. First of all, the camera position must be carefully selected. The camera must have an unobstructed field of view of all inspection features it is supposed to capture.

Also the camera's line of sight is not without difficulty. Only rarely will it be possible to capture all inspection features with an optimal angle. In most cases the

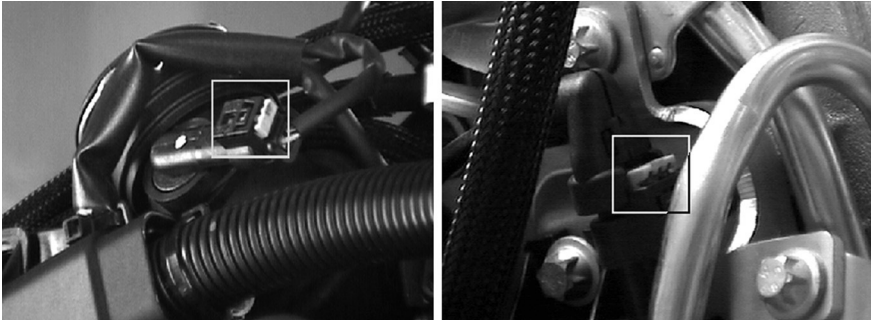


Fig. 9.1 Adverse angle of view and nearly obscured inspection feature

view will be slantwise. Therefore one has to make sure early on whether all inspection features can be checked from the planned position. In this example this does pose a problem for some inspection features. Some parts are partly obscured by other components while with others, the difference between correct and faulty fitting is almost parallel to the line of sight and therefore hardly visible. In Fig. 9.1, the sensor to be checked on the left side is hard to see because of the angle of sight, and on the right side one feature is almost obscured by a pipe.

When selecting the camera position, another aspect must be considered: the potential of the zoom lens. Since the lens is firmly integrated into the camera, it cannot be replaced with one having a different focal length. Thus both the maximum angle of view and maximum zoom are determined. Small inspection features must be close enough to the camera for the camera's resolution to be sufficient for inspection. Of course, an extension tube cannot be inserted behind the lens to shorten the minimum working distance. Therefore, the distance between the inspection features and camera must not be too short.

9.1.2 Illumination

The surface of the engine is curved in all directions and its components and the many inspection features tucked away necessitate illumination that is very diffuse coming from as many angles as possible. Individual illumination of each inspection feature would result in considerable effort because of their number. With a number of large-area light elements with fluorescent tubes, the engine can be illuminated diffusely from all angles so that almost all inspection features can be checked. For a few inspections it is better to switch off some elements of the illumination to improve contrast or avoid disturbing reflections.

9.1.3 Positioning

The engine is fastened in the inspection station so its position is fixed. However, it can be tilted around two axes. This results in position deviations in the images that also change from one perspective to the next. The rotation in the images can be neglected because of the engine's size, but even small angles can result in significant position changes of inspection features in the image. Since position changes are the larger the farther away the captured engine section is from the fixture, global position adjustment cannot be used but the deviation must be calculated for each image.

Since the inspection features are searched using template matching, one could try to look for the inspection feature directly within a sufficiently large region of interest without prior positioning. Unfortunately, even in detail view the size of the inspection features is still rather small. This necessitates a search with extremely small step sizes, which requires a lot of time. Furthermore, reflections occur on the engine's surface and that of its components that, in the worst case, can lead to matches with template matching thus to wrong segmentation.

Therefore, the position is adjusted. Searching for the reference for position adjustment also uses template matching with a large and uniquely shaped reference being selected such as the screw in Fig. 9.2 left. However, it does not necessarily have to be an individual component like a screw. A bended pipe can also be used as a reference as can be seen in Fig. 9.2 right. Naturally, the reference should be an object with a rather large area and little detail to allow for a large step size in the search.

9.1.4 Object Detection

As described above, the inspection objects are found using template matching. An obvious approach is to use the correctly fitted inspection object as the only

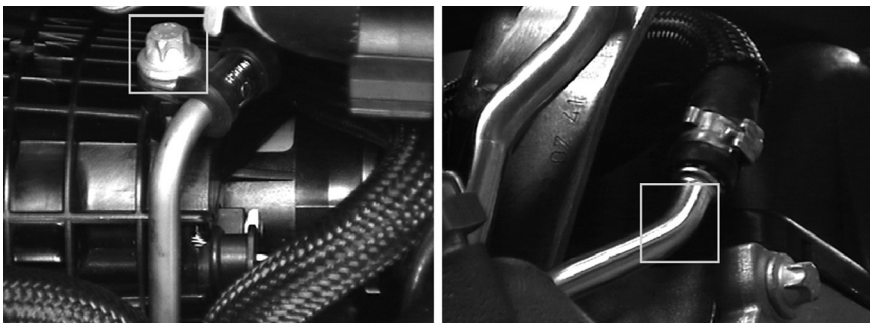


Fig. 9.2 Various position references

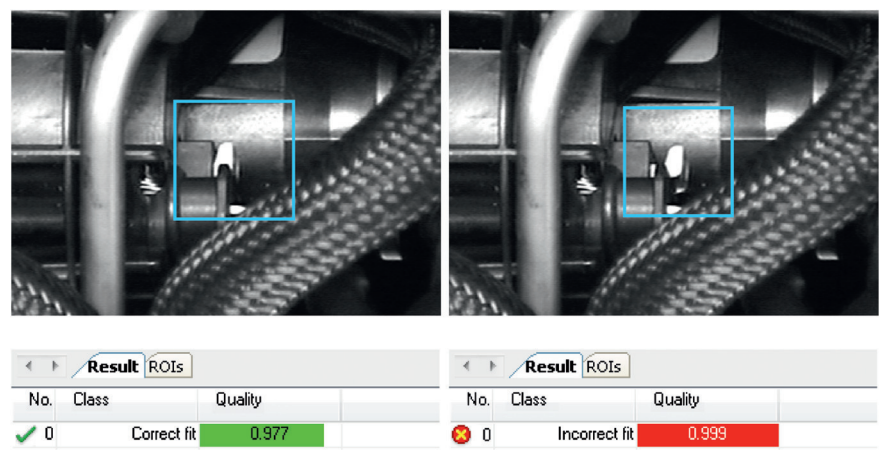


Fig. 9.3 Correctly and incorrectly fitted sensor

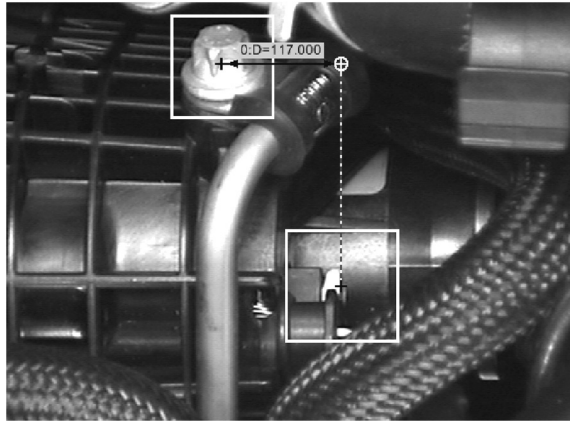
template. When these are found in the image, the degree of agreement between the template and the found object determines whether the inspection feature has been fitted correctly or not. This method works very well in a number of cases but there are inspection details where the difference between correct and faulty fitting causes only minor differences in correlation values. In these cases the search is extended by using not only the correctly fitted inspection object as a template but also the template of an incorrectly fitted one. Figure 9.3 shows a “correctly” fitted sensor on the left, and an “incorrectly” fitted one on the right. In both cases the object is found with sufficient correlation. Because of the template with which the sensor is found, we can clearly distinguish between correct and incorrect fitting; i.e. it is checked that the sensor is found with the “correct” template and a sufficiently high correlation. Else an error occurred since either the object is missing completely or incorrectly fitted.

Program hint: To remove objects because of their classification, NeuroCheck provides the check function *Evaluate Classes*. If *Verify target value* is enabled, you can specify a list of classes that will be accepted. All regions of interest (ROIs) of classes that are not on this list result in an error. In this case, the not OK class must not be in the list of acceptable classes so that regions of interest found with the not OK template are considered an error. *Delete pattern* causes ROIs determined as errors to be removed from the ROI list without causing the check function to be terminated with a target value error.

9.1.5 Verification of Results

Actually the inspection is already completed but let us imagine a case where the search for an inspection feature yielded a reflection. One way to prevent this is to

Fig. 9.4 Verification of gauging results



require higher quality for template matching. However, this can result in properly fitted sensors not being found. Therefore, we will take a different approach. To adjust the position, we have already looked for a reference in the image. The distance between this reference and the inspection feature is usually fixed with small tolerances. If the search for the inspection feature engages in a completely wrong spot, the distances between the inspection feature and position reference will deviate considerably from the normal values. Therefore, the distances in X and Y direction between position reference and inspection feature offer a good possibility to check the template matching results for plausibility. Basically, this will also check whether the position reference has engaged in the correct spot since this can also lead to an alteration of the distances between the inspection feature and position reference.

Since the screw head and sensor have already been found, this result verification merely consists of computing the distances of the two objects along the coordinate axes and comparing them to the usual distances; hence verification necessitates hardly any additional effort. Since we only have to verify that the correct objects have been found, neither gauging nor allowance check require high accuracy. An example for result verification can be seen in Fig. 9.4 where the distance in X direction between the sensor and the screw head—which incidentally is the position reference—is checked.

Table 9.1 shows the complete inspection program for one inspection feature with position adjustment, object search and result verification.

9.2 Simple Gauging for Assembly Verification

The example in this section demonstrates how to use methods from dimensional checking as a tool for presence verification. This is a typical method used in applications where the objective is to verify that a product has been assembled

Table 9.1 The following table summarizes the example from Sect. 9.1

Category	Description
Directory	\\Examples\\Presence
Illumination	Diffuse, large-area lighting
Functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, sets a region of interest in the surrounding area of the screw head 3. <i>Template matching</i>, searches for the screw head as reference for positioning the ROIs 4. <i>Determine position</i>, the found screw is used as position reference to compensate shifts because of engine tilts 5. <i>Define ROIs</i>, sets the ROI for the sensor search 6. <i>Position ROIs</i>, shifts these regions according to the position of the screw head 7. <i>Template matching</i>, searches the sensor with templates of correct and incorrect fitting 8. <i>Evaluate classes</i>, removes the found object if it was found because of using a template with an incorrect fitting or if the correlation quality between template and image is too low 9. <i>Count ROIs</i>, makes sure that an ROI exists, i.e. that a correctly fitted sensor was found 10. <i>Combine ROIs</i>, combines the position reference and sensor in one ROI list for gauging and verifying the result 11. <i>Gauge ROIs</i>, measures the distances between the reference and sensor in X and Y direction 12. <i>Check allowances</i>, verifies that the distances between reference and inspection object are correct and no faulty segmentation occurred

Fig. 9.5 Drip chamber for dialysis device

correctly, as in this example. The test piece shown in Fig. 9.5 is an IV drip chamber that is used as a blood filter in dialysis devices. Correct assembly of the blood filter behind the outflow of the chamber is of critical importance for function and safety of the product.

9.2.1 Illumination

Figure 9.6 shows the drip chamber under different lighting conditions. Obviously, inspection is possible only in transmitted light. Polarization improves definition of the edges. For the final realization, the area permitting light through the test piece was limited to the actually relevant inspection area to minimize problems caused by lateral illumination. Figure 9.8 shows the result.

The basic setup of the inspection system follows from the illumination requirements and the necessity to inspect the part from two different directions, which we will explain in the next section. Figure 9.7 shows a schematic top view

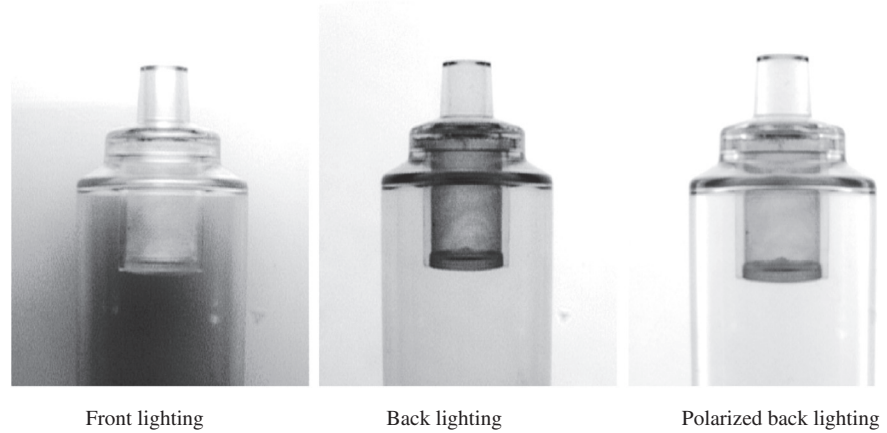
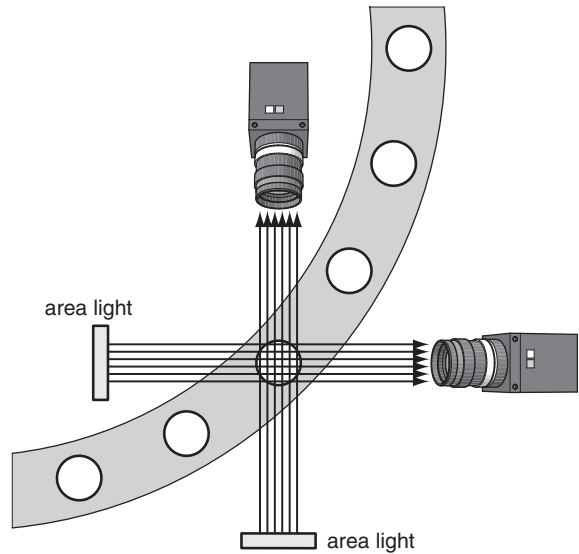


Fig. 9.6 Drip chamber in front light, backlight and polarized backlight

Fig. 9.7 Schematic structure of station for inspecting blood filter



of the setup. The drip chambers are transported upright on a conveyor belt through camera/light source combinations oriented at right angles so that an inclination of the filter can be detected independent of its direction.

9.2.2 Inspection Criteria

The most difficult stage in assembly verification in applications is often the specification of the criteria to be used for verifying correct assembly of a test piece. The intuitive human answer—“the part has to look right”—is unfortunately not very helpful. First, it has to be clarified what kind of assembly faults are possible and how they change properties of the test piece *in a quantifiable manner*.

In this example, incorrect assembly of the filter can manifest itself in the following ways:

- Skewed filter mesh, possibly causing a gap at the top edge of the filter.
- Deformation of the filter, leading to holes due to tensions in the material.

Figure 9.8 shows two images of a damaged filter captured at right angles. It can be seen that a deformation shortens the filter so that the distance between the bottom of the filter and the outflow of the drip chamber is changed. The skew of the filter causes the distances from the outer edges of the drip chamber to be different.

Figure 9.8 also explains why two images taken at right angles are required: taking only a single image, the system would miss a deformation or inclination of the filter in the direction of the optical axis of the camera.

From the possible defects of the filter listed above it follows that the following criteria have to be checked:

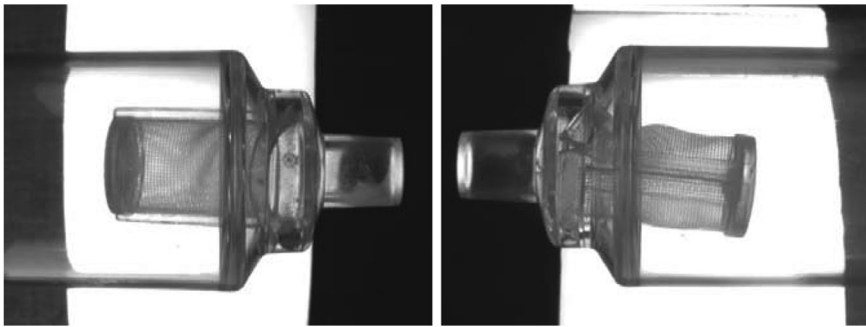


Fig. 9.8 Defective filter from two lines of sight with right angle offset

- length of the filter,
- centering within the drip chamber,
- diameter of the filter neck.

In the following we will show how to determine these properties.

Length of the filter: The length of the filter, i.e. the distance between filter bottom and outflow of the drip chamber, can simply be measured by segmenting the bottom part of the filter as one object, the top edge of the outflow as another. Figure 9.8 clearly shows that this edge is well defined by reflections inside the transparent plastic in front of the dark background. The edge as well as the filter bottom can be easily segmented by thresholding with individual thresholds for each object. Figure 9.9 shows the regions of interest used in the image from the first camera: a search line is used to probe the edge, whereas the filter bottom is segmented within a rectangular region of interest.

Centering: The subtask of checking whether the filter has been skewed during assembly might suggest measuring the direction of the outer edges of the filter. The angle of these edges may be altered by small irregularities in the outer contour due to inevitable—and acceptable—slight deformations of the filter. This is therefore not a suitable measure for the alignment of the filter. A much more reliable method for checking whether the filter has been skewed or bent is to measure the distances of the filter edges to the outer edges of the drip chamber. Since the objective is to determine correct centering, it is preferable not to test the individual distances against a specific tolerance, but instead to compare the difference of the distance in both directions to zero. This measurement is very stable and robust with respect to position and lighting variations. To determine the distance in both directions, four search lines can be used. Their arrangement is shown in Fig. 9.10. Along each search line a light–dark–edge is determined. The distances between corresponding pairs of edges are computed.

Diameter: In addition to these search lines, Fig. 9.10 shows two search areas used for measuring the diameter of the filter neck. Since the filter neck can in

Fig. 9.9 Search areas for creating filter bottom and outflow edge

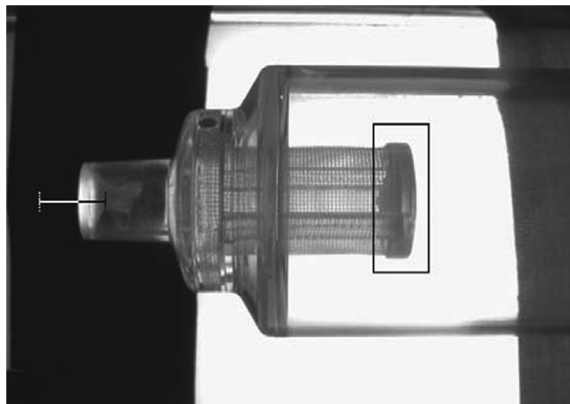
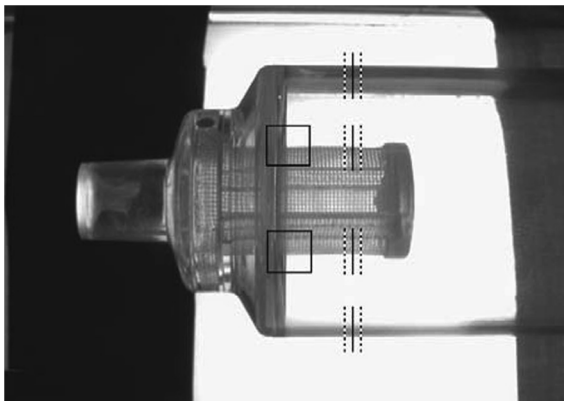


Fig. 9.10 Search lines for measuring filter orientation



practice not be separated visually from the body of the drip chamber, this diameter is measured indirectly by creating objects from the background areas limiting the filter neck on both sides. The distance of these background areas is equal to the diameter of the filter neck. One could call this a “negative measurement”: instead of measuring the width of the actual real-world object, the width of the interruption caused by the object in the background is measured.

9.2.3 Object Creation and Measurement Computation

All objects required to measure the criteria described in Sect. 9.2.2 can be segmented in one go. Because of the different brightness levels of the various object edges individual thresholds are needed in each region of interest for robust thresholding. Figure 9.11 shows the separately thresholded regions of interest within the otherwise unchanged gray level image on the one side, the segmented

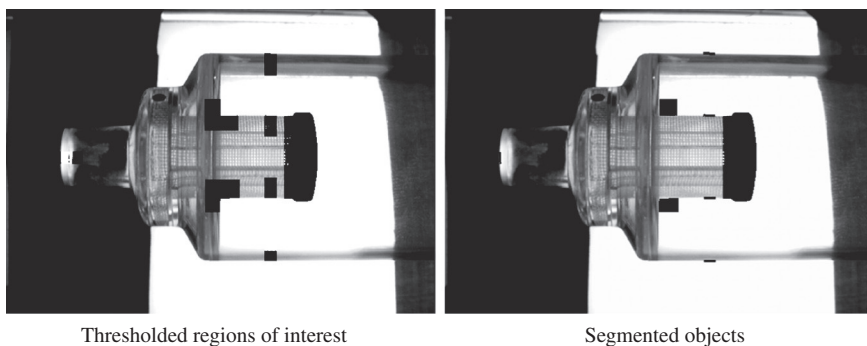


Fig. 9.11 Individually thresholded ROIs and segmented objects

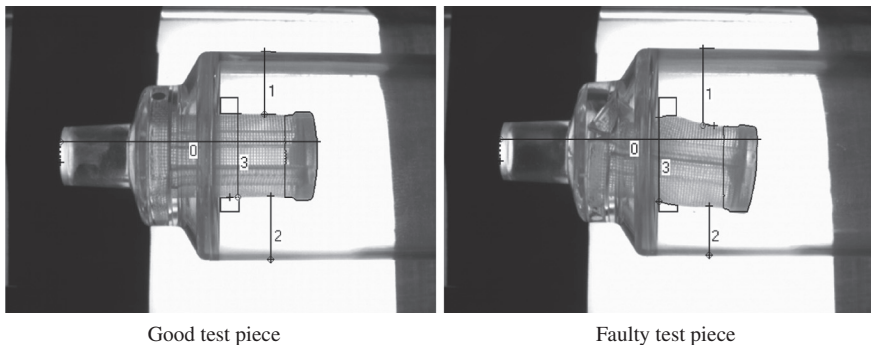


Fig. 9.12 Measurements necessary for inspecting a filter

objects drawn black into the gray level image on the other side (edge objects have been thickened for better visibility). Note the bright background areas created as objects above and below the neck of the filter.

In Fig. 9.12 the measurements used for the inspection have been drawn onto the images of a good test piece and a faulty one. The distance between the outflow and the filter bottom is measured in 0° -direction and is denoted by 0. All other measurements are taken in 90° -direction. It is important to specify the direction of the measurements to avoid measuring errors due to irregularities of the contours of deformed filters. Please note that the distances are measured in different places of the objects. This is because the minimum or maximum distances are measured. For the width of the filter (measurement 3) the minimum value is important, for the full length (measurement 0) the maximum value has to be taken to measure down to the filter bottom.

Finally the difference between the distances from the filter to the respective drip chamber edges labeled 1 and 2 in Fig. 9.12 has to be computed and compared with the pertaining nominal values. It is this measurement which leads to the rejection of the faulty piece shown in Fig. 9.12. It is larger by a factor of 25 than for a good piece.

9.2.4 Position Adjustment

We have left out one important aspect so far in this application. The test pieces are mounted on holders fixed on the conveyor belt. Due to the elasticity of the plastic, it frequently occurs that a test piece will not be precisely vertical on the holder. The inclinations will only amount to a few degrees, making an adjustment of the measuring directions of 0 or 90° unnecessary. They do have an effect, however, on the positioning of search regions. This is especially true for the search line creating the edge at the upper end of the drip chamber since, with a total length of 15 cm of

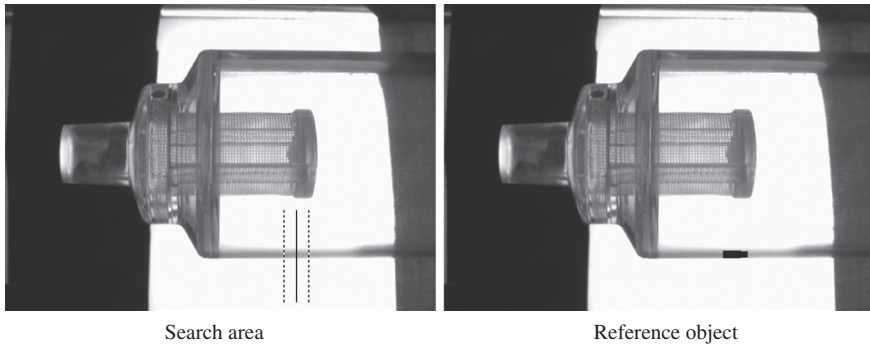


Fig. 9.13 Search area and reference object for position adjustment

the inspection part, a tilt of three degrees causes a deviation of 8 mm. Since the section to be inspected is rather short compared with the length of the inspection piece, rotation adjustment is not necessary, the ROIs can be repositioned using the position of a side edge. A prerequisite is the sufficient length of this search line. Figure 9.13 shows the search line and the generated reference object. We use the bottom edge as the reference object here because it is a very smooth edge which can be reliably detected.

Table 9.2 summarizes the entire inspection procedure for one of the two cameras. Apart from the different arrangement of regions of interest the checks for the two aspects are precisely identical.

9.3 Presence Verification Using Classifiers

The next example is taken from the automotive industry. We have clamping flanges that are secured on a shaft using a crimped nut. The application described in this section performs type identification and assembly verification on this test piece combining methods from basic presence verification, gauging and classification (Fig. 9.14).

The visual inspection of this test piece serves two purposes:

1. Ascertaining assembly of the correct flange type; the various types of flanges differ in the number of arms and in their length or in the distance of the bore holes from the center of the flange.
2. Verification of the crimping of the collared nut in the center of the flange. The crimping fixes the flange in a certain position on the shaft. If the nut is crimped incorrectly—for example because the crimping tool slipped—or not at all, the flange will not be fixed on the shaft.

Table 9.2 The following table summarizes the example from Sect. 9.2

Category	Description
Directory	\\Examples\\Presence
Illumination	Polarized back lighting with shields to avoid lateral illumination and diffraction effects
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, sets a search line for the front edge of the test piece (with respect to the movement of the conveyor) 3. <i>Determine threshold</i>, optimizes the threshold using a histogram analysis of the region of interest 4. <i>Create ROIs by thresholding</i>, segments a portion of the side of the drip chamber 5. <i>Determine position</i>, uses the bottom (or top, depending on the camera) coordinate of this section as position reference 6. <i>Define ROIs</i>, sets the regions of interest depicted in Figs. 9.9 and 9.10 7. <i>Position ROIs</i>, moves these regions according to the position of the edge of the drip chamber 8. <i>Determine threshold</i>, computes individual thresholds for each of the positioned regions of interest 9. <i>Create ROIs by thresholding</i>, segments the individual edges and sections 10. <i>Gauge ROIs</i>, computes the following measurements: • 0°-distance between outflow and filter bottom • 90°-distance between top edge of the drip chamber and top edge of the filter • 90°-distance between bottom edge of the drip chamber and bottom edge of the filter • Minimum 90°-distance between the background portions above and below the filter neck 11. <i>Derive measurements</i>, computes the difference between top and bottom distances between filter and chamber wall 12. <i>Check allowances</i>, compares the computed measurements with the nominal values

Fig. 9.14 Clamping flange with uncrimped nut



9.3.1 *Illumination*

As usual, the first question is that of suitable illumination. The decisive factor is rendering the crimping visible because the flange arms and the bore holes are much easier to find and identify than the crimping. Viewing the collared nut from the top, a human can easily discern crimped, uncrimped and incorrectly crimped nuts, but a reliable automatic identification requires much better contrast in order to have an existing crimping stand out clearly.

Correct crimping creates a surface area which is neither horizontal nor vertical in contrast to all other surface portions on the collared nut. This means that an illumination is needed which emphasizes differences in the inclination of surface areas, i.e. a directed illumination. It is clear that the crimping cannot be separated equally well from vertical and horizontal surface areas at the same time, but a good starting point for further inspection is the clear distinction of the horizontal areas on the one hand and the vertical areas together with the crimping.

Another requirement on the illumination follows from the rotational symmetry of the test piece and the fact that the location of the crimping along the perimeter of the test piece is unknown. It is therefore necessary to inspect the entire perimeter. This means that lighting also has to be rotationally uniform. Both requirements can be fulfilled by a large ring light illuminating the test piece obliquely from all sides. All horizontal areas appear extremely bright in the image, whereas the vertical edges of the nut as well as the crimping cast clearly defined shadows. Figure 9.15 shows the illumination effect schematically. The effect of the illumination is depicted extremely simplified to make the principle clear; the actual conditions are of course more complex due to the surface structure of the material and the rotational symmetry of part and lighting. For clarity's sake, reflected rays have only been drawn for three significant locations. The fanning out into several rays indicates that the surface is not an ideal mirror but scatters the light with a certain predominant direction. It can be seen that

- light from horizontal surfaces is reflected primarily in the direction of the camera. These areas appear bright;
- almost no light falling on rising, vertical edges is reflected towards the camera. These edges therefore appear dark;
- shadows appear behind falling vertical edges;
- only little light is reflected towards the camera from the inclined surface of the crimping, with the inclination presenting more working surface to the light than a vertical edge. This dark area therefore appears wider.

The effects of the lighting setup described above can be easily seen in the camera image. Figure 9.16 shows images of a crimped and an uncrimped nut. Because of the overexposure, the horizontal areas appear pure white, the vertical edges are visibly darker. However, they are disrupted in several places by the overexposure. The image of the crimped nut clearly shows the wide dark area of

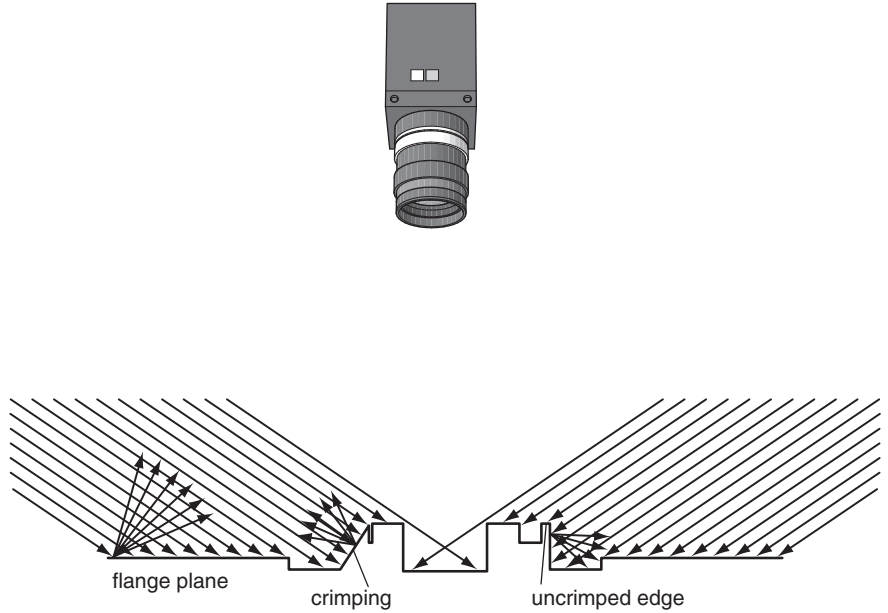


Fig. 9.15 Illumination of nut

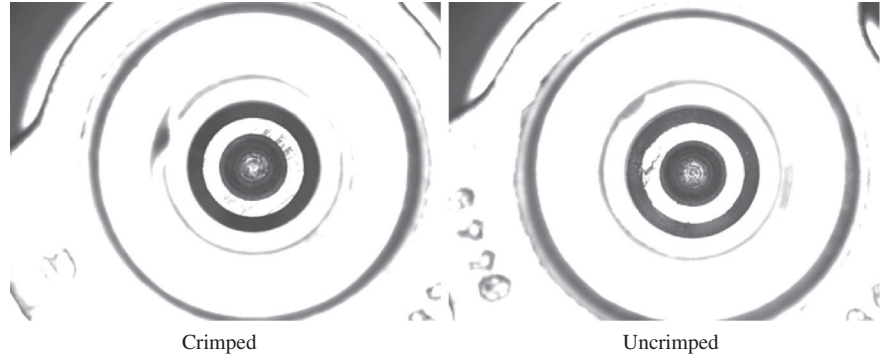


Fig. 9.16 Nut under ring light illumination

the crimping tapering out in correspondence to the shape of the crimping, whereas in the image of the uncrimped nut the edge appears only slightly thicker at the point where the crimping tool is applied. Incidentally, the illumination creates a wide shadowed ring, which is a very stable image feature that can be put to good use for positioning purposes.

This application is another striking example of the importance of illumination for image processing solutions. On the other hand, it also shows that the design of a suitable illumination setup requires at least some idea of how to realize the

inspection itself, because the algorithmic solution affects which features have to be visible in what way within the scene. We will later see that position adjustment is very important in this application. It is therefore very fortunate that the selected illumination method also creates a stable reference object for positioning in the image.

A word on the technical realization of the ring light: there are large ring-shaped fluorescent lights available but their fixed diameters limit flexibility and they do not produce entirely uniform all-around lighting because there has to be a starter at some place. For these reasons, a ring light was constructed from separate LED spots, which have the additional advantages of great durability and high yield at wavelengths for which CCD sensors are especially sensitive.

9.3.2 Check of Crimping

Preprocessing: The vertical edges, which appear as thin dark lines in the camera image, cannot be used for positioning or other evaluation purposes because they may be partly interrupted due to the overexposure. The exact location of these interruptions cannot be determined beforehand because the test piece may not be positioned precisely in the center of the ring light. For the inspection they are therefore a disturbance whose influence can be reduced by preprocessing with a dilation filter as can be seen in Fig. 9.17.

Determining of position reference: The next step is the definition of a region of interest within the outer borders of the collared nut. Within these limits, thresholding based on a histogram analysis can be performed easily, taking into account that a considerable portion of the pixels is concentrated at gray level 255 due to the overexposure. The suppression of artifacts by removing extreme gray levels from the calculation described in Sect. 4.2.3 is therefore not recommended in this application. Fig. 9.18 shows the defined ROI and the corresponding

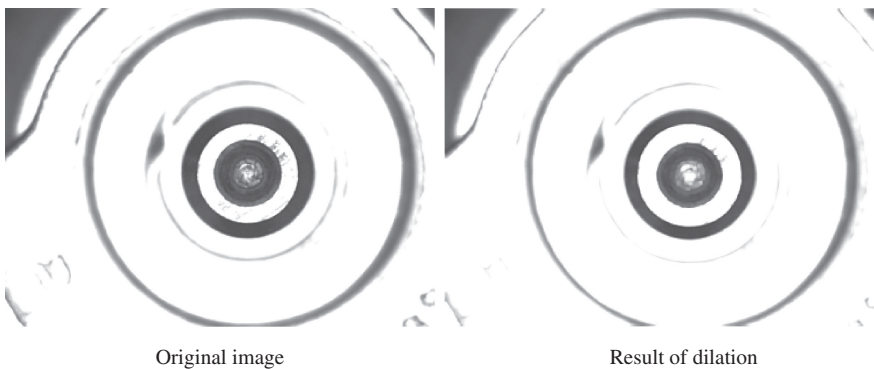


Fig. 9.17 Nut before and after gray level dilation

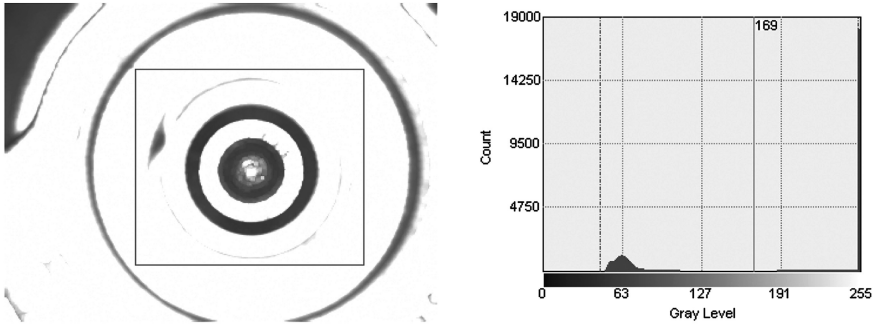


Fig. 9.18 Region of interest for positioning around nut (black rectangle) and corresponding histogram

histogram. As is customary for applications with position variations, the first region of interest has to be large enough to find the relevant objects for the maximum position variation. On the other hand, one will choose the smallest region possible in order to reduce the time required for the segmentation and the number of additional objects which would have to be screened out later. The wide shadow ring in the middle of the nut can be easily segmented as reference object for the position determination. Parts of the vertical edges not suppressed by the dilation as well as the crimping itself may be detected as additional objects. The object sizes are sufficiently different to suppress these spurious objects already during segmentation. For safety reasons, a validation using features significantly different for the spurious objects and the shadow ring like perimeter, area, form factor and ratio of width to height is recommended. Figure 9.19 shows this on the example of an uncrimped nut where all objects have been detected. Perimeter and area are much larger for the shadow ring than for the spurious objects, as is the form factor; the ratio of this almost perfectly circular object is always close to one.

It turns out that the inner border of the shadow ring is smoother and more stable than the outer one; therefore, another segmentation is performed inside the circular object for a bright object. After another validation stage similar to the one described above, this inner circle is used as the position reference.

Unrolling: What is the purpose of this complex procedure of determining a reference object? The main issue of this inspection task is the insecurity when it comes to the angle position in which to find the crimping. Thus the entire perimeter has to be searched.

On the other hand, Fig. 9.19 shows that there can be several spurious objects at almost equal distances from the center of the nut which are very difficult to distinguish from the crimping using simple features. This means that one should immediately search for an object of the appropriate shape, which suggests using template matching. The object's pixel shape is changed depending on the angular position of the crimping, leaving two possible methods:

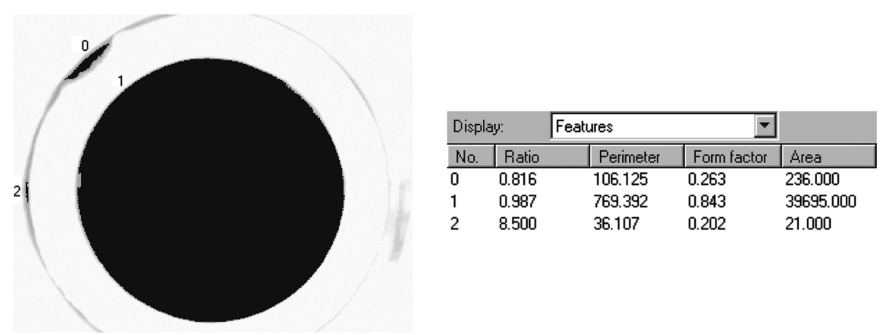


Fig. 9.19 Shadow ring with spurious objects and corresponding features

- Creating (or internally computing) separate templates for a multitude of angular positions. On the down side, the computational effort for correlation increases significantly with the number of templates. It has to be taken into account that several templates are necessary since the crimping does not always have the same shape.
- The circular ring inside of which the crimping has to be located can be converted to a rectangular image. Thanks to the large radius of the circle, discrete effects are limited and the computation effort is restricted to a single conversion.

To perform such a conversion of a ring into a rectangle, a circle line is defined around the center of the ring. At certain angular increments the normal on this circle is computed. Then the gray levels to both sides of the circle are interpolated parallel to this normal direction and written column-wise into a new, rectangular image. Figure 9.20 shows the definition of a circle along the shadow border of the collared nut and the result of the unrolling (to enhance visibility, contrast has been improved and the overexposed background has been shaded in gray).

This method can in principle be applied to arbitrary contours, but it is advantageous to use a manually defined circle because then size and shape are exactly specified. It is of special importance to position the circle very accurately. Figure 9.21 shows the effect of unrolling when the center of the defined circle and center of the collared nut do not coincide. The unrolled image shows a noticeable wave which may change the shape of a correct crimping as to be unrecognizable.

There is one other problem with the unrolling process: in contrast to a circle, unrolling requires a defined starting and end point. If this falls in the middle of the crimping, unrolling will cut it in two parts whose shapes differ from the original unbroken shape making further segmentation impossible. It is immediately obvious that this can be avoided by “over-unrolling” the circle by 50 %. This means that the first 180° are appended a second time to the end of the image. It is then absolutely certain that the unbroken crimping will be present at least once in the unrolled image.

Fig. 9.20 Definition of a circle and result of unrolling

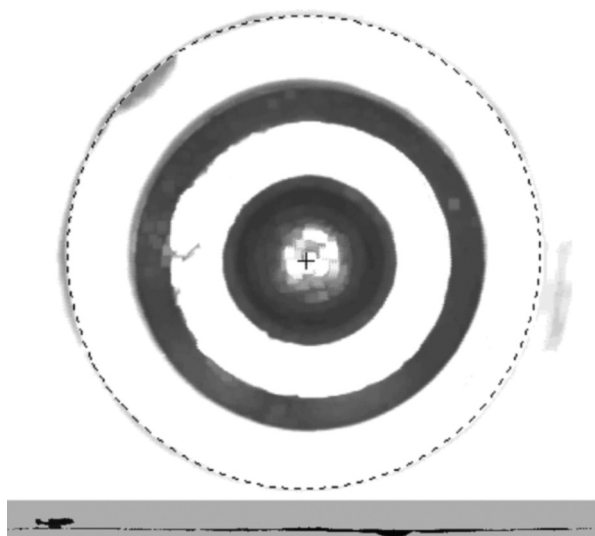
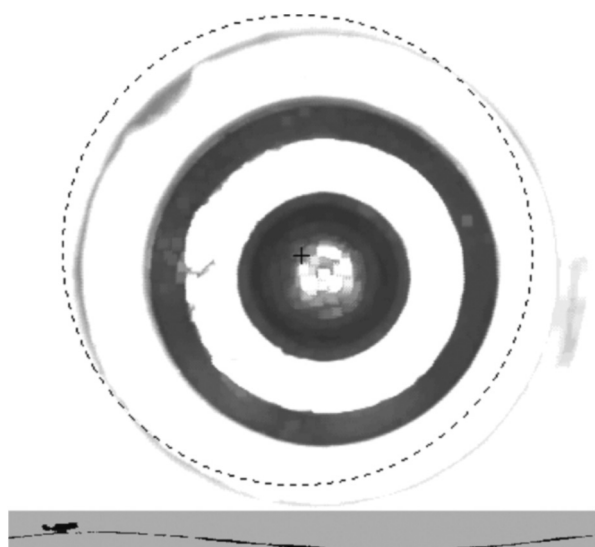


Fig. 9.21 Definition of a circle and result of unrolling with incorrect positioning



Segmentation and evaluation of the crimping: Because of its compact shape, the crimping can be segmented easily and reliably using template matching. It is recommended, however, to include images of incorrect and missing crimping in the template set and to optimize the positions of the correlation points according to the shape of the crimping. Figure 9.22 shows some templates.

But the inspection task is not solved after finding the crimping using template matching. The main capabilities of template matching are in the area of segmentation, not classification. It is therefore quite easy to locate a good crimping,

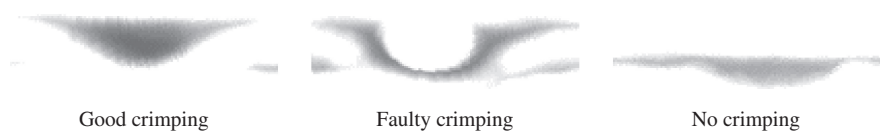


Fig. 9.22 Templates for searching the crimping

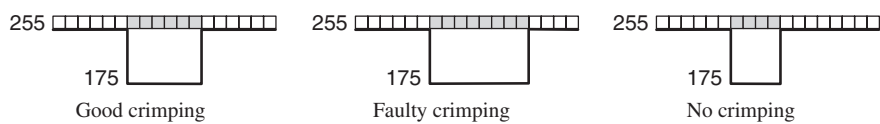


Fig. 9.23 Distinction problems with template matching

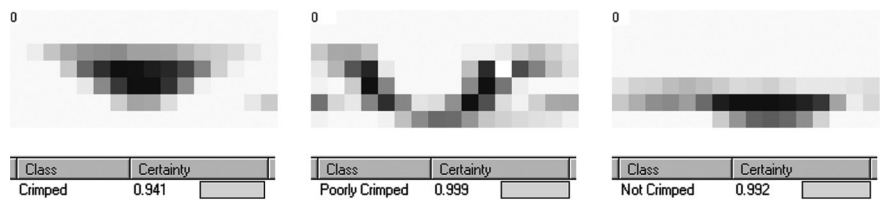


Fig. 9.24 Resampled and classified crimping

but it is much more difficult to distinguish it from a faulty crimping or the point where the tool is applied. A one-dimensional example may help to illustrate this problem. Figure 9.23 shows schematic cross sections of the crimping. For simplicity reasons let us assume that there are only two gray levels: 255 and 175. The narrow dark area with a width of only four pixels corresponds to the point where a tool is applied, which is a bit wider than normal, whereas the wide dark area represents a faulty crimping.

When computing the correlation between the good crimping and the other two cases in the sample profile, one obtains a correlation coefficient of 0.79 with the faulty crimping and of 0.76 with the uncrimped point where the tool is applied. In reality, cross sections will not be quite so similar. But because of the considerable variation in the shape of the crimping, it is not possible to require high minimum correlation coefficients, so it will be difficult to distinguish good, faulty and missing crimping on the basis of the correlation coefficient alone. It is therefore necessary to evaluate the detected object again, for example by presenting its appearance to a classifier. Figure 9.24 shows the result of such a classification using a resampled image of 16×7 pixels and the average gray level inside the bounding rectangle of the object. This average gray level will be significantly lower for correctly crimped parts because the dark area of a correct crimping is much larger than that of a faulty or missing one.

Table 9.3 The following table summarizes the example from Sect. 9.3.2

Category	Description
Directory	\\Examples\\Presence
Illumination	Low, incident ring lighting from LED lights with strong overexposure
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Filter image</i>, performs a dilation to remove thin edges 3. <i>Define ROIs</i>, sets a rectangle in the approximate center of the flange 4. <i>Determine threshold</i>, optimizes the threshold using a histogram analysis of the region of interest 5. <i>Create ROIs by thresholding</i>, segments the shadow circle in the center of the flange. The object can be validated by check functions Compute features and Screen ROIs 6. <i>Create ROIs by thresholding</i>, segments the bright inner area of the shadow circle, which is better suited as position reference because of its more stable contour. Again the object can be validated by check functions Compute features and Screen ROIs 7. <i>Determine position</i>, uses the centroid of the bright inner area as position reference 8. <i>Define ROIs</i>, creates a circle with the exact diameter of the edge where the crimping is located 9. <i>Position ROIs</i>, centers this circle according to the centroid of the bright inner area 10. <i>Unroll ROI</i>, transforms a strip to both sides of the positioned circle into a long, rectangular image 11. <i>Define ROIs</i>, declares the entire unrolled image as the search area 12. <i>Template matching</i>, scans the unrolled image for the crimping 13. <i>Compute features</i>, calculates the average gray level inside the bounding rectangle of the detected crimping 14. <i>Resample ROIs</i>, creates an input image for the classifier from the detected crimping 15. <i>Classify ROIs</i>, determines whether the detected candidate is a good crimping from the average gray level and the resampled image

This solves the first part of the inspection task, the verification of the crimping. Table 9.3 summarizes the example. We can see that it is a rather time-consuming inspection. Decisive factors for the inspection time are the functions *Filter Image* with the computational intensive dilation, *Unroll ROI* with the interpolation along the circle and *Template Matching*, which has to use a small step size here to allow for a sufficiently reliable detection of the varying and rather ill-defined shape of the crimping.

9.3.3 Type Verification of the Flange

Now we turn to the second part of the inspection task, checking the type of the flange. The various types of flanges differ in the number of arms and their length.

Fig. 9.25 Three-armed flange

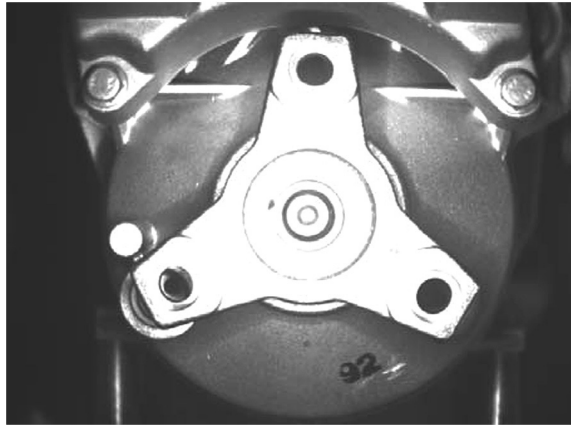


Figure 9.25 shows a complete three-armed flange. We can see a correctly crimped nut in the middle. It can be seen that the flange can be inspected using the same illumination as for the crimping. Only the camera has to be switched because a much larger field of view is required for this part of the inspection.

The entire flange is difficult to segment as it can merge with background areas, depending on its angular orientation. It is much simpler and more reliable to segment sections from the flange arms, count these, and gauge the distance between the center of the collared nut and the bore holes in the flange arms to check the length of the arms. A high level of accuracy is not necessary in this application since only a type distinction is required.

The inspection starts in almost the same way as that of the crimping in Sect. 9.3.2, with the following alterations:

- Small spurious objects from the edges of the nut are not important in this inspection; the computation intensive dilation filtering can therefore be left out.
- The inspection is less position-sensitive, therefore the slightly less smooth and stable outer contour of the shadow circle suffices as reference object.

The final configuration is almost identical to that in Sect. 9.3.2 before unrolling. Again, a circle is defined running through the roots of the flange arms slightly outside the base of the flange itself, as Fig. 9.26 shows. A binary thresholding along this circle that restricts contour tracing to a narrow area on both sides of the circle segments a section from each arm without the time-consuming unrolling. For the correct flange this results in three objects also shown in Fig. 9.26.

As usual, several features are computed for each of these objects to ascertain that they are actually sections from the flange arms and not undefined objects from the background. These features have to be independent of the angular direction of the flange arms. Some suitable features are, for example, the area and the maximum radius, i.e. the maximum distance between the centroid of the object and its contour. After all objects have been removed which do not comply with the criteria

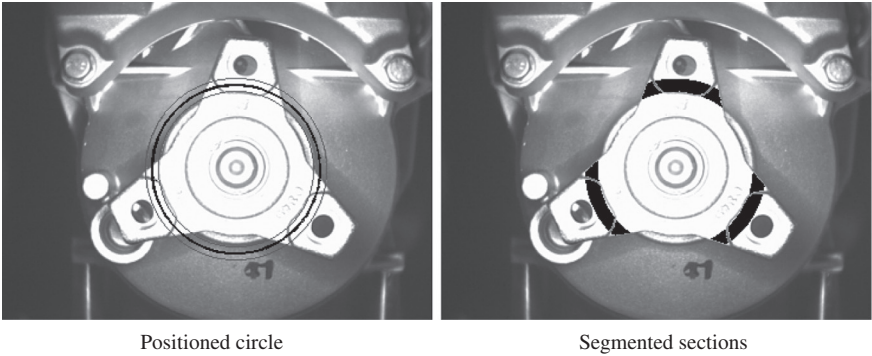


Fig. 9.26 Flange with positioned circle and segmented sections of the arms

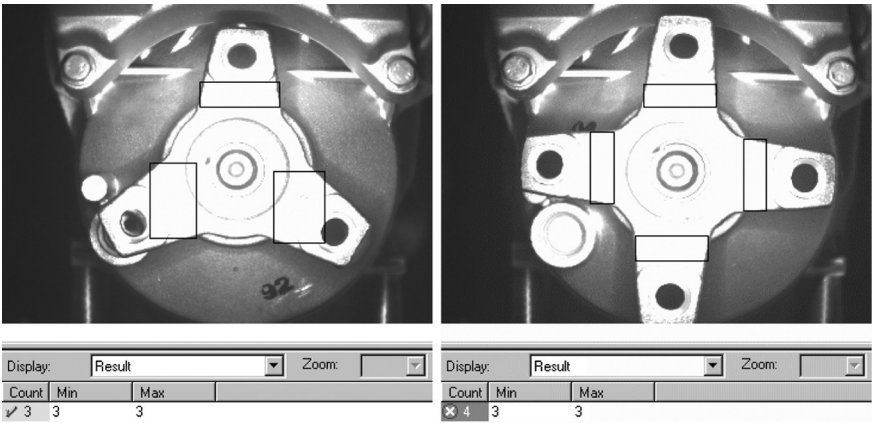


Fig. 9.27 Flange with three and four arms

for flange arm sections, the remaining objects are counted. If the number of objects does not equal three at this point, the check is aborted immediately because it cannot be the correct flange. Figure 9.27 shows both cases, a correct three-armed flange and a flange for which the sections of the arms meet all criteria—except for the fact that four arms are present.

Presence of the correct number of arms does not necessarily mean that the correct type of flange is also present. For this to be true, the length of the arms also has to be correct. This is determined by measuring the distance between the center of the flange and the bore holes since the bore holes can be segmented more reliably than the far edges of the arms, which may merge with bright background areas depending on the angular orientation of the flange. The bore holes are such clearly defined objects that they can easily be found by template matching without

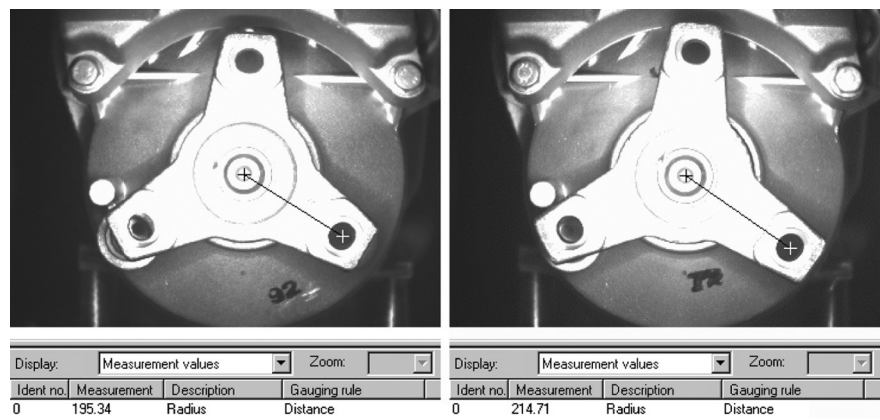


Fig. 9.28 Flanges with varying arm lengths

particular regard to the region of interest. The search area can simply be defined as a square encompassing the entire flange. It is also not necessary to restrict the template search to a circle defined around the center of the flange. Although this would limit the area to be searched, the computational complexity would be increased due to the more complex addressing of the positions to be searched so that there would be no significant increase in speed.

Since this application does not require particularly accurate gauging but only a relatively coarse distinction between various types, one can simply search for a single bore hole and measure its distance to the center. For the same reason, no calibration to metrical measurements is required. The scale will be identical for each flange, so one can directly relate the pixel-based measurement values. Figure 9.28 again shows two cases. Both flanges are three-armed—else the check would have stopped before this point—but the arms of the right flange are too long.

Finally it should be noted that a precise measuring of the flange diameter would be done a little differently. In this case all three bore holes should be detected by template matching, then the exact contours of the bore holes created by thresholding inside the template areas. As we know from Sect. 7.2.1, these centroids are subpixel-accurate. A model circle can then be computed according to Sect. 7.5.1—which for the three defining points in this case is an exact circle but still subpixel-accurate because it is based on the subpixel-accurate centroids. The radius of this model circle is a very accurate and stable measurement for the radius of the circle of bore holes. Prerequisite for this method is that all bore holes are visible as complete, unbroken circles without irregularities in their contours. Table 9.4 summarizes the example.

Table 9.4 The following table summarizes the example from Sect. 9.3.3

Category	Description
Directory	\\Examples\\Presence
Illumination	Low, incident ring lighting from LED lights with strong overexposure
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, sets a rectangle in the approximate center of the flange 3. <i>Determine threshold</i>, optimizes the threshold using a histogram analysis of the region of interest 4. <i>Create ROIs by thresholding</i>, segments the shadow circle in the center of the flange as position reference. The object can be validated by check functions Compute features and Screen ROIs 5. <i>Determine position</i>, uses the centroid of the shadow circle as position reference 6. <i>Define ROIs</i>, creates a circle running through the bases of the flange arms 7. <i>Position ROIs</i>, centers this circle according to the centroid of the shadow circle 8. <i>Create ROIs by thresholding</i>, cuts out sections from the flange arms along this circle as new objects 9. <i>Compute features</i>, computes characteristic data (area and maximum radius) from the sections of the flange arms 10. <i>Screen ROIs</i>, removes all objects with invalid feature values leaving only actual flange arm sections 11. <i>Count ROIs</i>, determines whether the flange has the required number of arms 12. <i>Define ROIs</i>, sets a search area encompassing the entire flange 13. <i>Position ROIs</i>, positions the search area again according to the centroid of the shadow ring 14. <i>Template matching</i>, searches for a single bore hole 15. <i>Gauge ROIs</i>, measures the distance between the centers of bore hole and flange 16. <i>Check allowances</i>, verifies the compliance of the measured distance with the prescribed size of the required flange type

9.4 Contrast-Free Presence Verification

This section is about inspecting the tube holder of an air-conditioning unit in Fig. 9.29. Rubber tubes are wedged between the individual noses during assembly. It can happen that some noses are not molded correctly or break off later. The main difficulty with this subtask is that the noses are molded together with the part from

Fig. 9.29 Tube holder on air-conditioning unit



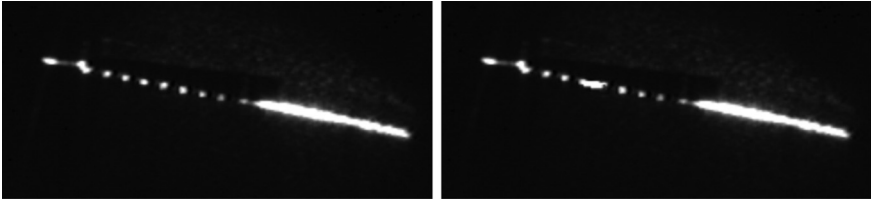


Fig. 9.30 Proper and faulty tube-holder with laser line illumination

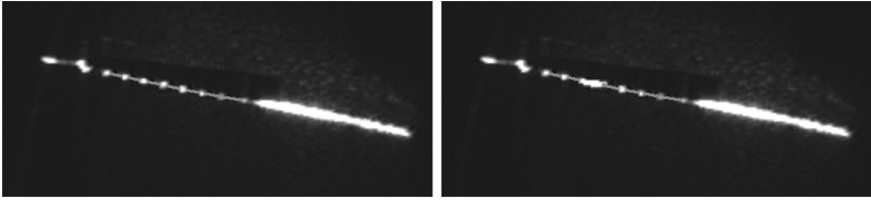


Fig. 9.31 Interspace objects for interval check at hose holder

the same dark plastic. It is therefore practically impossible to design an illumination yielding reliable and sufficient contrast.

Such a problem can be solved by structured lighting. Figure 9.30 shows the effect of projecting a laser line onto a good tube-holder and a tube-holder with a missing nose. It can be clearly seen that in the second case one of the required shadows is missing.

To inspect the tube-holder, all bright objects, i.e. the reflections of the laser line on the dark plastic, are first created using thresholding. All objects are screened out which do not correspond to the typical appearance of such a reflection. Since the reflections are small, almost circular objects, features like area, maximum radius and form factor, i.e. the normalized ratio between area and perimeter, can be used. Then the remaining objects can be counted.

In addition to directly counting the objects one can also check the spatial frequency, for example by connecting the center points of the objects with a line and performing another segmentation along this line, this time for dark objects. By restricting this segmentation to the line connecting the center points the dark gaps between the bright reflections are created as objects. These can be checked again for their length and counted. Both checks together determine the presence of the correct number of objects in the correct intervals, thus preventing an additional reflection outside of the area of the tube-holder to convey the impression that an actually missing nose was present. The objects created in this interval check are indicated in bright gray between the reflections in Fig. 9.31.

Program hint: A line connecting the center points of objects is automatically created in **NeuroCheck** when a model line is created for the entire group of objects using check function *Compute Model Geometries*. The check function creates a virtual object from the control points as an intermediate step for the

calculation of the regression line. These are just the centroids of the objects in question. The complete check can be found in the download in directory \Examples\Presence\Airco.

Finally, it should be noted that such structured lighting using laser lines also allows for actual gauging—sufficient resolution provided—as shown in Sect. 8.6.2.

9.5 Presence Verification Using Line-Scan Cameras

All examples from the previous sections used an area-scan camera image for inspection, i.e. a rectangular image that was captured at once at a certain time. This method reaches its limits with certain types of inspection pieces. This is the case, for example, with very large objects where the fixed sensor resolution of an area-scan camera usually doesn't allow for sufficiently detailed images. Another very common example is cylindrical parts that occur in a great variety in all parts of industry. We are first going to comment on when the inspection of lathe parts is possible using area-scan cameras and what difficulties can be encountered.

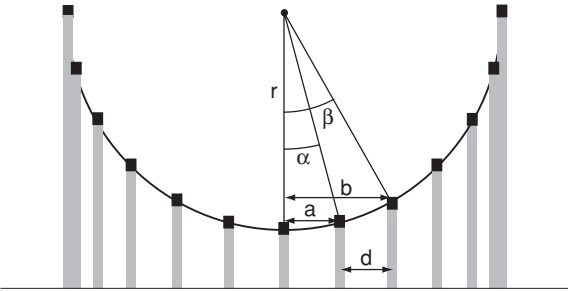
9.5.1 *Inspection of Cylindrical Parts with Area-Scan Cameras*

In the previous examples we've already dealt with some visual inspections of rotation-symmetric parts. Section 7.6 shows gauging of the outside contour of a thread flange, Sect. 9.2 the assembly inspection of a drip chamber. The film rolls from Sect. 5.4 and the spark plug from Sect. 1.6 are further examples for inspections of cylindrical parts.

In all these examples the inspection features either referred to the cross-section of the inspection parts (as with the thread gauging) or had to be determined through individual images of certain sections of the part surface (as with character recognition on the spark plug or block code recognition on film rolls). Such inspection tasks involving round parts can be solved using the "ordinary" image of an area-scan camera. However, if the inspection task involves capturing the entire surface of a round inspection part, using an area-scan camera poses several problems.

Shortening of projection: The main difficulty is that the part's surface is curved away from the object. When projecting the curved surface onto the sensor plane, more or less distinctive shortenings occur outside the image center. The broader the section of the part captured in the image, the more pronounced the shortening. In the extreme case that the entire part is captured in the image, the shortening on the outermost edges is infinite, i.e. each arc length on the surface shrinks to a length of zero in the image. Figure 9.32 shows the effect of this projection with the

Fig. 9.32 Shortening of objects along a circle line when projected onto a sensor plane



assumption that the camera is focused on the axis of a lathe part. On the circumference of this part, objects are arranged with a 15° distance. One can see that when projecting onto a planar sensor, the distance of the objects to each other gets smaller to the outside. The outermost objects even fuse together. If, for example, the objects are lines of a bar code, this code wouldn't be legible.

If two objects on the circumference of a cylindrical part are located at the angle positions α and β , according to Fig. 9.32, their distance d in the projection onto the sensor plane is:

$$d = b - a = r(\sin \beta - \sin \alpha). \tag{9.1}$$

The (not depicted) distance d_k of objects on the circumference of the circular cross section is just equal to the arc length between the two angle positions:

$$d_k = 2\pi r \frac{\beta - \alpha}{360^\circ}, \tag{9.2}$$

with the angles in degrees. Table 9.5 gives an impression of the degree of shortening occurring with growing distance from the central view direction. Between 90° and 75° , an arc length of 26.18 mm is shortened to 3.41 mm in projection from an object with 100 mm radius.

Task types: An essential criteria for being able to solve an inspection task involving a cylindrical part with an area-scan camera is how far the area to be inspected covers the circumference of the part, and whether it is possible to present the relevant areas directly, if possible, in the line of sight of the camera.

Table 9.5 Shortening of distances when projecting the circumference of the circle onto a planar sensor

Radius	Alpha ($^\circ$)	Beta ($^\circ$)	Arc	Projection	Remaining length (%)
100	0	15	26.18	25.88	98.86
100	15	30	26.18	24.12	92.12
100	30	45	26.18	20.71	79.11
100	45	60	26.18	15.89	60.70
100	60	75	26.18	09.99	38.16
100	75	90	26.18	03.41	13.02

A perspective shortening can already be seen in the introductory example character recognition in Sect. 1.6, Fig. 1.9 but which was rather limited so close to the middle of the projection.

Also of importance is whether the inspection feature is axial or radial oriented. The bar code on the film rolls in Fig. 5.29 is far away from the middle of the projection, the scanning, however, is done at a right angle so the shortening is of no importance for reading the bar code.

Basically, inspecting a cylindrical object with an area-scan camera is thus possible if one of the following conditions is met:

- The inspection features only refer to the contour or cross-section of the object (as with the thread gauging in Sect. 7.6 and assembly inspection in Sect. 9.2).
- In the projection on the sensor plane, the inspection features are so close to the part's axis that the shortening because of the curvature is very small (as with the digits on the spark plug in Sect. 1.6).
- The inspection features are prominently axial oriented so the radial shortening is of no great importance (as with the bar code on the film roll in Sect. 5.4).

Furthermore, fluctuations in the orientation of the inspection piece in front of the camera (that lead to varying optical shortening of inspection features) have to be limited to a range that is permissible for each inspection task. Even when the characters are along the axis, character recognition would certainly be difficult or impossible if the inspection piece is turned so far that the characters are only visible along the edge.

Gauging tasks: Special problems pose themselves when gauging on the surface of cylindrical objects since the shortening changes the geometric resolution of the image and thus calibration and accuracy depending on the angle position. Table 9.5 shows that already in a range of $\pm 15^\circ$ on both sides of the direct line of sight of the camera, distortions of more than one percent occur. With increasing distance from the optical axis, this distortion grows rapidly because of the non-linearity of the angle function. Strictly speaking, measuring such a piece with an area-scan image is only possible with the part's contour or along the axis, not along the circumference.

Inspecting the complete circumference: When the inspection features cover the entire circumference of the inspection piece and an area-scan camera is to be used, several images have to be captured. The number of images depends on the required resolution and permissible distortion. If, for example, a shortening of a maximum of ten percent is permissible, then no more than $\pm 30^\circ$ of the part's circumference can be captured in one image according to Table 9.5. Thus six images are necessary for each part. These six images can be captured in various ways:

- Several cameras are arranged around the part. This way is rather expensive (because of the required number of cameras) and also poses illumination problems since the illumination for one camera can blind other cameras.

- The part is turned and captured cyclically, i.e. in each of the desired angle positions, in front of the camera. This solution requires a high mechanical effort and complex control for the step motor and trigger signals. Since the part is again and again stopped and set back in motion, image capturing takes a lot of time which is aggravated by vibrations that can occur and for whose quieting down one has to wait.
- The part is continuously rotated in front of the camera. This is mechanically more simple than a clocked rotation, but poses even more serious synchronization problems since the individual image captures have to be triggered in specific angle positions. Since the part is captured in motion, exposure can only be very short to avoid blurring. This, in turn, results in the problem to get enough light onto the curved surface in the moment of image capture.

One can see that an inspection covering the entire circumference of a cylindrical part cannot be solved satisfactorily using an area-scan camera. The inspection of lathe parts is a typical line-scan camera application.

9.5.2 *Inspection of a Valve Body*

Figure 9.33 shows a section of a valve body used to control the oil flow in automotive hydraulics. This image shows the illumination problems posed by these parts. Under a certain angle, the curved, polished surface reflects so strongly that a homogeneous illumination over a large area is rather difficult.

Let us take a closer look at two of the inspection criteria for these components:

- The bore holes in the lathed cut-ins of the part are vital for the proper oil flow. Their number and size have to be checked.
- The raised contact surface between the lathed cut-ins must be free of cavities since they have to seal the oil flow.

Fig. 9.33 Valve body from automotive hydraulics



Image capture: A line-scan camera is used for inspection. As described in detail in Sect. 8.4, image capturing with a line-scan camera always necessitates relative movement between inspection object and camera. Figure 8.15 shows the image capturing situation for capturing the surface of a cylindrical component using a line-scan camera. The component surface is scanned line by line with the camera, i.e. unrolled into a rectangular image.

The capturing method is fundamentally different from capturing a rotating component using an area-scan camera.

Rotation: Since the line-scan camera scans the component surface continuously line by line, the component is continuously rotated and not stopped.

Synchronization: If the rotation is sufficiently even, the image processing system or the camera only need a trigger signal for the start of the rotation. Only when very high accuracy with regard to gauging and geometry is required, synchronization of rotation and image capturing via rotary encoder according to Sect. 8.4.4 is necessary.

Distortion: Since the camera only captures one line at a time along the component axis, the camera always looks down on the part's surface at a right angle so there are no curvature distortions.

This way we can obtain a distortion-free unrolling of the component surface. The necessary time for such image capturing results from the required geometric resolution (see Sect. 8.4.2), the required exposure time (see Sect. 8.4.3), and the mechanical conditions that determine the speed with which the component can be rotated. For the application described here, image capturing takes about 3 s.

Illumination: The inspection tasks mentioned above, bore hole and surface inspection, require different illuminations:

Bore hole inspection: For the inspection of the oil flow openings, a diffuse bright field illumination is used. The polished surface strongly reflects the light so the dark bore holes contrast sharply.

Surface inspection: Spurious cavities in the contact surfaces can be detected with a directed dark field illumination. The principle of the illumination is shown in Fig. 5.26. The even, smooth surface reflects the directed incident light past the camera while cavities on the surface disperse the light in the direction of the camera. For the illumination a cold light source with line projector was used. Capturing and illumination angle must be adjusted to each other.

Figure 9.34 shows the line-scan camera images of the inspection part with the described illumination conditions. All in all, two captured images are necessary for the inspection with different illuminations. It is advantageous to capture both images in sequence at the beginning of the inspection routine instead of capturing first one image and processing it completely. The images can be buffered and processed while the component is leaving the inspection station. This way, part of the handling time can be used for inspection.

Bore hole inspection: Because of the good contrast in the bright field image, the bore holes can be easily detected using thresholding. However, accurate positioning is important to narrow down the search for the bore holes to a restricted area. This can reduce the risk to find the edges of the cut-ins that also appear dark in the image.

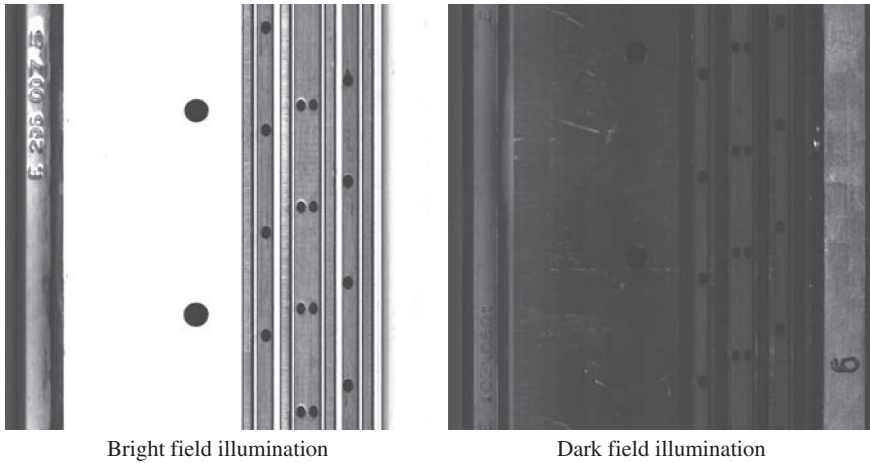
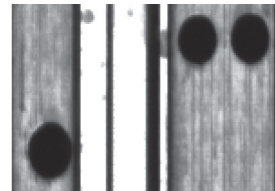


Fig. 9.34 Line-scan camera images of the valve body with various illumination conditions

Fig. 9.35 Reference template for position determination in bright field



This way, there are also less spurious objects that need to be filtered; the more artifacts are present, the more difficult it can be to find good filter criteria. Also, it is quicker to search a smaller area. The characteristic group of three holes depicted in Fig. 9.35 offers itself as a reference object for position determination.

Further inspection takes place in the following steps:

Search areas: Depending on the position of the first bore hole group found, the search areas for the bore holes are set. Please note that we are only concerned with the horizontal position. The vertical position is undetermined anyway since no information is available concerning the angle position of the component with which the line-scan camera started capturing. A full revolution of the part is captured starting from the position of the part when it enters the station.¹

Segmentation: The bore holes are segmented within these search areas using a threshold. Template matching as for positioning wouldn't suffice. For the

¹ Actually more than 360° are unrolled to avoid cutting off a bore hole in the image because it is at the starting point of the revolution. With line-scan camera applications it's a common approach to capture slightly more than a full revolution.

Table 9.6 The following table summarizes the example from Sect. 9.5.2

Category	Description
Directory	\\Examples\\Presence
Illumination	Diffuse front lighting, bright field
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, sets a search area for detecting the reference position 3. <i>Template matching</i>, detects characteristic bore hole group as position reference 4. <i>Determine position of ROIs</i>, measures the offset from the reference position 5. <i>Define ROIs</i>, sets a search area for segmenting the bore holes 6. <i>Position ROIs</i>, adjusts the search areas according to the determined deviation 7. <i>Determine threshold</i>, optimizes the bright/dark-threshold using a histogram analysis of the search areas 8. <i>Create ROIs by thresholding</i>, segments dark objects within the search areas 9. <i>Compute features</i>, computes characteristic values of found objects 10. <i>Screen ROIs</i>, eliminates spurious objects 11. <i>Screen ROIs</i>, checks agreement of remaining objects with characteristic features of bore holes 12. <i>Count ROIs</i>, checks whether the correct number of bore holes is present 13. <i>Gauge ROIs</i>, computes the distances of neighboring bore holes within a cut-in 14. <i>Calibrate measurements</i>, converts pixel measurements into metric ones 15. <i>Check allowances</i>, checks the distances between neighboring bore holes

subsequent inspection of the bore holes’ sizes, it is necessary to determine the actually visible contour.

Removal of artifacts: After thresholding the spurious objects are removed that may result, among others, from surface structures and the edges of the cut-ins. For this reason all objects are removed that are not sufficiently circular (measured using the height-width ratio and form factor; see Sect. 10.1.2) and whose area is not within certain limits.

Inspection of bore hole size: Subsequently, the inspection checks whether each cut-in shows the required number of objects of the proper size. Since the component is rotated more than 360° in front of the camera, one object more is admitted than bore holes are present.

Inspection of geometric arrangement: Finally, the distance between two neighboring bore holes in the same cut-in is checked to make sure there is no bore hole in the wrong spot; this would, after all, not manifest itself in a faulty number.

Table 9.6 summarizes the check routine for the bore hole inspection.

Inspection of contact surfaces: Please note that the example in this section is highly simplified. Especially with applications dealing with surface inspection, time-consuming and elaborate experimental runs are necessary to capture the variety of possible error types, and also the variety of appearances of faultless surfaces. It’s not unusual for an application engineer to have to “think outside the box” to achieve a reliable distinction between OK and not OK parts using pre-processing, masking known surface effects and so on.

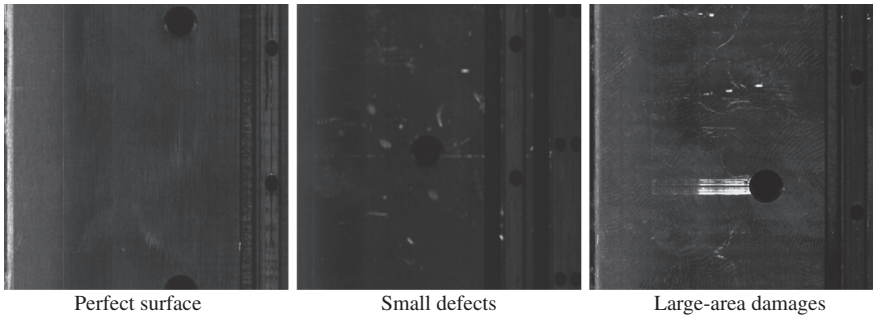


Fig. 9.36 Various valve bodies with dark field illumination

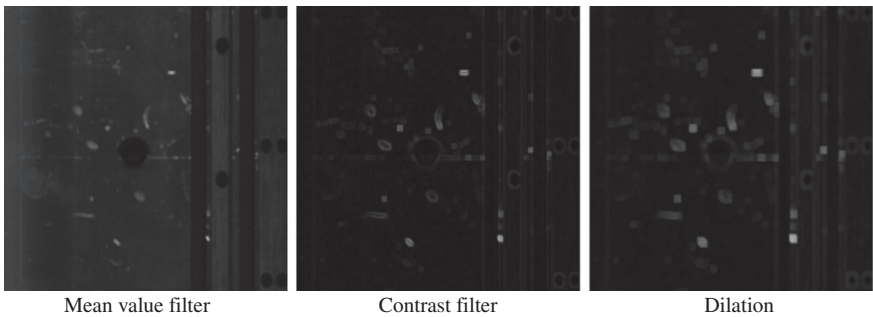


Fig. 9.37 Valve body with damaged contact surface after preprocessing

Figure 9.36 shows segments of the surface of three different valve bodies. The inspection part in the left image has a perfect contact surface; the part in the middle has a number of small defects while the part on the right has large-area damages. A surface inspection of this type where certain individual defects are searched can be compared to presence verification except you're checking for the absence of defects.

Here, the defects need to be enhanced first so they can be reliably detected. For this, the image is filtered three times:

Mean value filter: This low-pass filter is used to smooth tiny machining artifacts that could be confusing in the subsequent screening stage.

Contrast filter: This non-linear filter improves the inhomogeneous nature of the gray level distribution caused by the defects.

Dilation: To improve detection, the bright defects are thickened.

Figure 9.37 shows the middle image from Fig. 9.36 after the three filtering stages.

After preprocessing, the bright defects are searched for in the contact surface area. They are evaluated according to area and brightness. Finally, the remaining bright objects are counted. If no objects remain that were determined to be defects,

Table 9.7 The following table summarizes the contact surface inspection from Sect. 9.5.2

Category	Description
Directory	\\Examples\\Presence
Illumination	Directional front lighting, dark field
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i>, sets a search area over the entire contact surface 3. <i>Filter image</i>, mean value filter for smoothing machining artifacts 4. <i>Filter image</i>, contrast filter for enhancing defects 5. <i>Filter image</i>, dilation to thicken bright objects 6. <i>Determine threshold</i>, optimizes the bright/dark-threshold using a histogram analysis of the search areas 7. <i>Create ROIs by thresholding</i>, segments bright objects within the search area 8. <i>Compute features</i>, computes characteristic values of found objects, in this case the maximum brightness and the area 9. <i>Screen ROIs</i>, leaves objects fulfilling the criteria for a defect according to size and brightness 10. <i>Count ROIs</i>, determines the inspection piece as “not OK” if an object, i.e. a defect, remains after object filtering

the entire inspection piece is deemed correct. Table 9.7 summarizes the check routine for the contact surface inspection.

9.5.3 Notes

In conclusion we would like to make a few remarks on the basic properties of line-scan camera applications.

Image size: Usually, line-scan camera applications deal with very large images. In the above examples we have images of merely 1,000 × 1,000 pixels that could also be captured using an area-scan camera. However, image sizes of more than 50 million pixels are not unusual.

Necessary time: Line-scan inspection of components usually require a lot of inspection time. This is caused on the one hand by the processing time for large line-scan camera images; on the other hand by the necessity to move camera or inspection part to get a contiguous image. The necessary time can be reduced by capturing the next image while the previous one is being evaluated; thus, these two time-consuming steps can be executed in parallel.

Surface inspection: This is a very typical application area for line-scan cameras. In the present example, the surface was inspected using presence verification after appropriate preprocessing. This is a method that can often be used, especially if it’s important that a surface is free of certain defects.

If, however, the surface is required to have a certain structure, methods from texture analysis need to be used that are much more computation-intensive than the preprocessing filters employed here. Protracted optimization processes for a

large number of inspection parts—and especially from batch production—are typical for applications of the surface inspection type since the appearance of technical surfaces and the possible defects are manifold. It is therefore advantageous if the image processing system used allows for a fast, interactive optimization of the running application.

Illumination: Like area-scan camera applications, line-scan camera inspections depend on adapted illumination. Since the camera only captures a very small section of the image scene, a single line, line-scan camera applications are very sensitive to adjustments in illumination and camera. On the other hand, the areas that need to be illuminated are so small that homogeneous lighting is easier to achieve.

Geometry: For line-scan camera applications that depend on the unrolling of the surface of a cylindrical part, the vertical position of objects in the image is usually undetermined. It depends on the angular position of the inspection part with which the image capture was started. An appropriate position compensation has therefore be taken into account.

Program hint: In **NeuroCheck** the check function *Adjust Line-Scan Image* can be used utilizing a determined position in the image to make sure that the image structure is always identical.

9.6 Summary

The following list summarizes some general properties of presence verification problems. Since this application area is rather heterogeneous, inspection routines can seldom be built according to predefined structures. Hardware setups (illumination, number of cameras...) are correspondingly varied. It did not seem useful to include such special aspects of the applications described above in this summary.

Presence verification problems: This is a very heterogeneous application area, ranging from simple object counting to validation of basic features and recognition of objects using classifiers to assembly verification with gauging methods.

Simple presence verification: In this type of application the validity of the objects to be inspected can usually be checked using simple feature screening. By this, we mean checking features of individual objects using numerical limit values. The typical procedure is as follows: after segmentation, basic features are computed such as size, brightness, etc., invalid objects are removed, remaining objects are counted. If the segmentation is performed by template matching, the correlation coefficient is already a first measure for the validity of detected objects.

Assembly verification with gauging: In this type of application, validity of objects follows from gauging results; i.e. in addition to basic features, relationships between objects, e.g. distances, angles, circle similarity, etc. are important. It is usually more important how the measurements relate to reference objects than achieving particularly high accuracy.

Presence verification using classifiers: If complex interdependencies between features have to be checked to determine the validity of objects, or if the entire appearance of the objects is important, classifiers can be used for object screening which have been trained to distinguish appearance and features of valid and invalid objects.

Contrast-free presence verification: Structured lighting, for example laser lines, can be used to segment relevant objects under conditions where no visual contrast can be achieved—for example in the case that the foreground, i.e. relevant objects, and background are made out of the same material.

Relevant properties: The main difficulty in most presence verification tasks is translating the quality criteria into measurable, computable quantities. Example: the requirement that a filter must not be mounted askew can be checked most reliably by measuring symmetrical distances.

Chapter 10

Overview: Object Features

Most examples in the preceding chapters used various features to check image objects for validity. It is therefore high time to give an overview of such features. From the numerous features described in literature, we have selected some that have proved their worth in many industrial applications. At the same time, we will present some further insights into the difficulties encountered when applying common everyday notions to the discrete world of digital images.

10.1 Basic Geometric Features

We summarize values such as coordinates, simple dimensions, area etc. under basic geometric features. The distinction between geometric and shape-describing features we are making here derives mainly from algorithmic complexity and is, of course, in part arbitrary.

10.1.1 Enclosing Rectangle

Every segmented object can be enclosed by a rectangle. In the simplest case this rectangle is oriented along the coordinate axes. The following basic features can be derived from this:

Origin: The coordinates of the top left corner of the enclosing rectangle correspond to the x coordinate of the leftmost point of the object and the y coordinate of the topmost point of the object contour.

Dimensions: The *width* or x dimension of the rectangle is equivalent to the difference of the x coordinates of the far right and far left points of the object contour. Correspondingly, the *height* or y dimension is equivalent to the difference of the y coordinates of the topmost and the bottommost points of the object contour.

Ratio: The ratio between height and width of the enclosing rectangle. This way we have a very simple, shape-describing parameter.

As simple as these features may appear, they can be very useful in practice. As we have seen in the character recognition applications in Chap. 5, segmentation algorithms will not necessarily find objects in the sequence expected by a human observer. If characters are to be read line by line they will therefore have to be sorted by x and y coordinates. Usually the origin of the enclosing rectangle will be sufficient for this sorting operation and can be computed much faster than the objects' centroid—or may even be available from the segmentation algorithm anyway.

It should be noted that for objects found by template matching, the enclosing rectangle is identical to the object border. Since an object segmented by template matching does not need to show a closed, visually identifiable contour, its limits can only be determined from the size of the template.

Oriented enclosing rectangle: A more elaborate definition for the enclosing rectangle uses the direction of the principal axis of the object as coordinate axis. In this new coordinate system the enclosing rectangle can be defined in the same way as above.

Rectangularity: Russ (2007) and Sonka et al. (2008) define the *rectangularity* or *extent* of an object as the ratio of the object area A to the area A_{ser} of the smallest enclosing rectangle.

$$R = \frac{A}{A_{ser}}. \quad (10.1)$$

This feature is 1 for an exact rectangle, otherwise smaller. For an exact computation of this feature the oriented enclosing rectangle has to be used. Else, an exact, but rotated rectangle would give a much smaller value than 1.

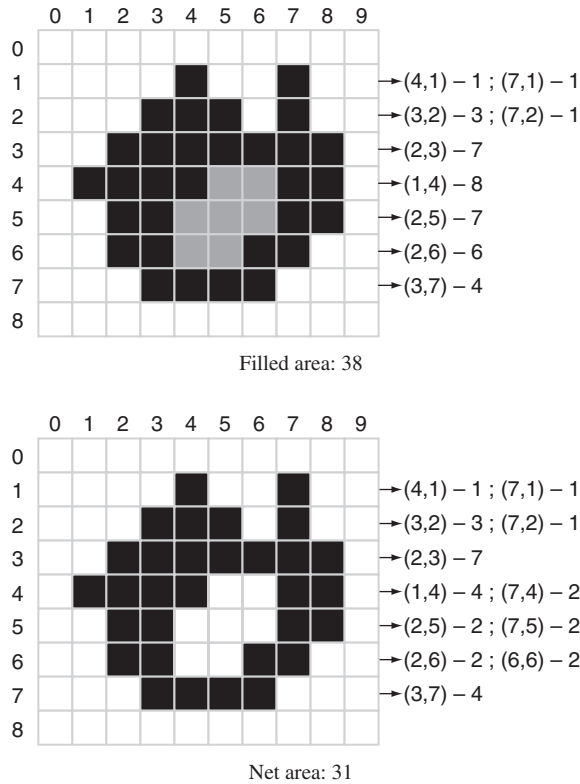
10.1.2 Area and Perimeter

In general, features like area, perimeter and the like are meaningful only for objects segmented using thresholds or other brightness properties since objects found by template matching do not necessarily correspond to a visually closed image structure. In the following, we will therefore always assume that objects are represented as contours.

Area: The *area* of a segmented object is very easy to compute. It is equivalent to the number of pixels inside the object contour. The area of a thresholded object can be defined in two different ways:

Enclosed area: The enclosed area encompasses all points inside the object contour including those that would not belong to the object according to the threshold.

Fig. 10.1 Run length codes for area computation



Net area: For the net area all regions within the object that fall on the other side of the threshold are not included in the area computation.

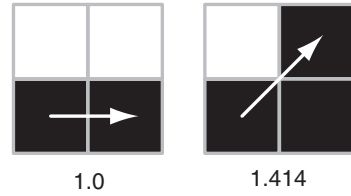
Area computations are often performed using a run-length representation of the object. Each line of the object is represented as a series of segments, or *runs*. All the pixels in each run belong to the object, i.e. runs do not contain non-object pixels. Every run segment is described by its starting point and its length. The area of an object is then simply the sum of all run lengths. Figure 10.1 demonstrates the run-length encoding of an object featuring an indentation as well as a “hole”. After each line the corresponding segments are written out with their starting points in parentheses followed by the length of the segment.

Program hint: The **NeuroCheck** check function *Create ROIs by Thresholding* provides the “Fill regions” option. If this option is switched off, objects are created without holes, i.e. a subsequent area computation will result in the net area.

Perimeter: The *perimeter* of a segmented object can be computed directly from the chain code. Some caution has to be applied, though, because one cannot simply count the pixels of the contour:

1. The true contour runs neither along the outer edges of the contour pixels nor along their inner edges. Instead, it is located at some position in between, which

Fig. 10.2 Perimeter contribution of contour pieces



depends on the combined effects of spatial discretization and gray level quantization, and is usually assumed to be in the center of the pixels.

2. The distance between the center points of two pixels depends on the angle of the contour at that point, as shown in Fig. 10.2:
 - A vertical or horizontal connection between two pixels contributes a length of 1.0 to the perimeter.
 - Diagonal connections contribute a length of $\sqrt{2} \approx 1.414$ to the perimeter.

Taking the above properties into account, it is very simple to compute the perimeter from the chain code. From the definition of the chain code in Sect. 4.3.3 it follows that all diagonal contour pieces are encoded by odd values of the chain code, all vertical and horizontal segments by even values. The perimeter of the approximate circle in Fig. 10.3 is then given by:

$$P = 8 \times 1.0 + 8 \times 1.414 = 19.312.$$

An exact circle running through the centers of the contour points would have a radius of three pixels, resulting in a perimeter of

$$P = 6 \times \pi = 18.85$$

pixels. Despite the small number of pixels, the deviation is only approximately 2.5 %. Nevertheless, perimeter values have to be used with caution. They depend strongly on the image resolution. With increasing magnification, more and more irregularities will become apparent in the contour, increasing the perimeter.¹ Furthermore, perimeter values are extremely noise-sensitive. Figure 10.4 shows the results of perimeter and area computations on circles with and without additive noise. Adding noise renders the contour lines more irregular, causing the perimeter to increase by up to one-third, whereas the area differs by no more than 3 %. This does not mean, however, that the perimeter is a completely useless feature. Industrial image scenes contain mainly smooth edges, so that the perimeter can be applied quite well to characterize objects.

¹ This is the root of the coast line paradox according to (Mandelbrot 1987): a coast line will become longer as the ruler becomes shorter. The behavior of object properties for different degrees of magnification is the subject of a separate branch of image analysis using fractal dimensions.

Fig. 10.3 Perimeter calculation of discretized circle

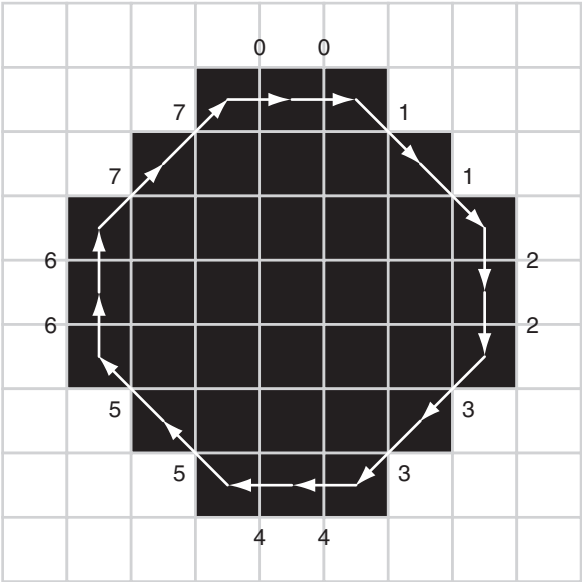
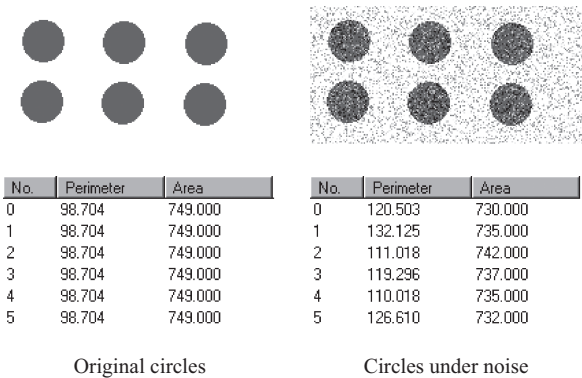


Fig. 10.4 Effects of noise on area and perimeter calculations



Form factor: An interesting feature can be computed from area and perimeter, the form factor. We will introduce it here because of its close connection with the previous two features, but it could as well be categorized into the group of shape-describing features since it gives identical results for identical geometrical shapes, independent of scale—apart from the inevitable discretization effects of digital images.

Nischwitz et al. (2007) define *form factor* as the ratio of perimeter squared to area. This measure has the minimum value possible for a circle since a circle with a given perimeter encompasses the maximum area of all geometric shapes. Form factor would thus be better defined the other way around. Since a circle is the most compact of all shapes, it should have the highest value. Therefore, we define the form factor as the ratio of the area to the squared perimeter according to Russ (2007):

$$F = \frac{4\pi A}{P^2}. \quad (10.2)$$

The ratio is scaled by 4π so that the circle with its area of πr^2 and its perimeter of $2\pi r$ has a form factor of precisely 1. Any other geometrical shape has a smaller form factor.

10.1.3 Centroid Coordinates

The *centroid* (or center of gravity/mass) of an object is defined as the point that would serve to balance the object. The coordinates (x_c, y_c) of the centroid can be computed by averaging the coordinates of all points of the object over its area A :

$$x_c = \frac{\sum_i x_i}{A}, \quad (10.3)$$

$$y_c = \frac{\sum_i y_i}{A}, \quad (10.4)$$

The centroid can also be computed directly from the contour. This has the advantage that less points have to be used in the computation, making it faster. On the other hand, extremely irregular portions of the contour exert an overly strong influence on the centroid because they consist of a disproportionately large number of pixels. According to Russ (2007) this can be corrected by using the following formulas:

$$x_c = \frac{\sum_i (x_i + x_{i-1})^2 \cdot (y_i - y_{i-1})}{A}, \quad (10.5)$$

$$y_c = \frac{\sum_i (y_i + y_{i-1})^2 \cdot (x_i - x_{i-1})}{A}, \quad (10.6)$$

$$A = \frac{\sum_i (x_i + x_{i-1}) \cdot (y_i - y_{i-1})}{2}. \quad (10.7)$$

The cost for the advantage of having to use less pixels lies therefore in an increased computation effort for every point.

We have already mentioned in Sect. 7.2.1 that the coordinates of the centroid can be computed to subpixel accuracy thanks to averaging over the entire object. Actually, the centroid is one of the most stable features that can be computed for an object.

10.1.4 Axes and Radii

Principal axis of inertia: We have already presented the definition of the principal axis and methods for its computation in Sect. 3.2.1, therefore, we will repeat only the most important aspects here.

The principal axis of an object is defined as the axis through the center of gravity around which the object has its lowest moment of inertia. This is equivalent to demanding that the sum of the distances of all pixels belonging to the object from the axis be minimal. Formal calculation according to Eq. (3.1) requires very high computational effort for larger objects. Therefore, various approximation methods have been developed.

Parker (1994) describes an approximation method based on the assumption that the axis passes precisely through a contour pixel of the object. Since the axis has to pass necessarily through the centroid, one can proceed as follows:

1. Mark all pixels on the object contour above the centroid.
2. For each line through the centroid and each of these pixels compute the mean distance of all pixels belonging to the object from this line.
3. The line with the smallest mean distance from all pixels can be used as an approximation of the principal axis.

Enclosing polygon: By rotating the coordinate axes stepwise, one obtains a new enclosing rectangle for each new coordinate direction. According to Russ (2007), the points where these enclosing rectangles touch the contour can be used as the vertices of an enclosing polygon. An angular increment of 10° yields nine different orientations of the axes and thus 36 vertices. The axis of this polygon can again be used as approximate principal axis.

Maximum Feret's diameter: Russ (2007) defines *Feret's diameter* as the length of the projection of an object on a coordinate axis. Width and height of the enclosing rectangle are therefore the Feret's diameters of an object in the standard coordinate system. The maximum Feret's diameter is equivalent to the maximum distance of two points on the object contour. Although the direction of this longest chord through the object does not constitute an approximation of the principal axis, it is nevertheless related to the orientation of the object—if one disregards discretization effects due to the limited image resolution. Furthermore, the length of this chord can itself be used as a simple characteristic of the object's shape.

Radii: By *radius* we denote here the distance from the object's centroid to any point on the contour. By scanning the complete contour, the minimum and maximum radius and their respective orientations can be found. Furthermore, an average radius and other statistical properties of the set of all radii can be computed. All these values can be used to describe the object's shape and its orientation. For circular objects the maximum and minimum radii are also determined. The differences between these radii are mainly caused by the discretization of the image; therefore, they are not related in a systematic way. However, well-defined shapes with a known predominant direction can be clearly distinguished using these parameters.

Storing every single radius value in a vector results in the same distance-versus-angle signature used in Sect. 3.2.2 to determine orientation. In addition to being a very stable measure for the object orientation, it can also be used to identify objects according to the correlation of their signatures. At this point we have reached the—somewhat arbitrary—dividing line between dimensional features and shape descriptors, which we will discuss in the following section.

10.2 Shape Descriptors

10.2.1 Contour Curvature

An object contour can be treated as a planar curve. The curvature of a planar curve is defined as the rate of change of slope, i.e. the change of the tangent direction. Bronstein et al. (2005) give the following relationship between the tangent directions α and α_0 at points P and P_0 , and the arc length $\overline{PP_0}$ between these points for the curvature:

$$C = \lim_{P_0 \rightarrow P} \frac{\alpha - \alpha_0}{\overline{PP_0}}. \quad (10.8)$$

Figure 10.5 illustrates this relationship.

It can be shown that for a curve defined by $y = f(x)$ the curvature takes the form

$$C = \frac{y''(x)}{(1 + y'^2)^{3/2}} \quad (10.9)$$

Formal calculation of the curvature therefore requires the first and second derivative of the contour. Digital contours typically exhibit a certain “irregularity” due to discretization effects. These effects are similar to adding noise, i.e. variation, to a smooth curve. By definition, differentiation amplifies variations. It is therefore quite difficult to obtain reliable values for the curvature of a contour in this way.

Fig. 10.5 Formal definition of curvature

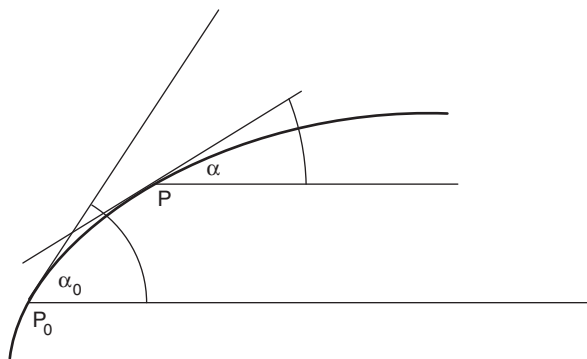


Fig. 10.6 Approximation of curvature

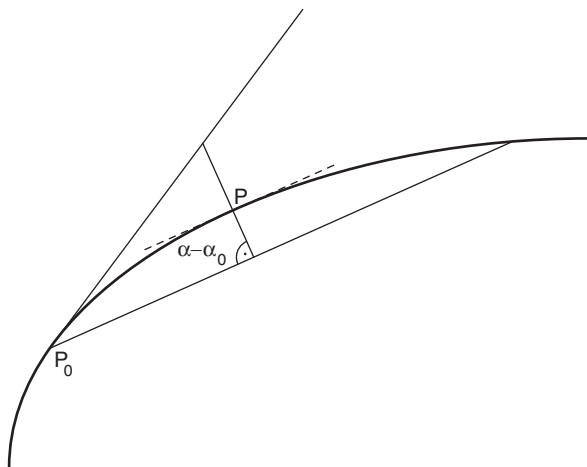
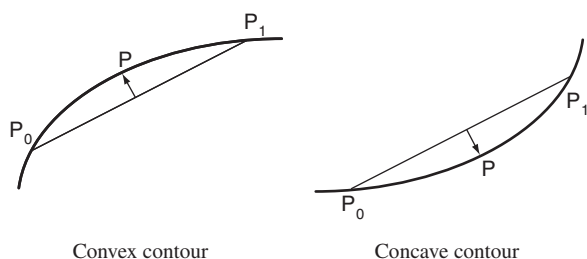


Fig. 10.7 Curvature in convex and concave contours (chord runs from left to right)



Moving the tangent at point P so that it intersects the curve at point P_0 shows that the perpendicular distance between this line and the tangent of point P_0 , measured at point P , is related to the difference of the angles $\alpha - \alpha_0$ as can be seen in Fig. 10.6. The angle will open further with increasing curvature, thereby also increasing the distance between the two lines.

Computing the tangent direction is equivalent to computing a first derivative of the curve. Because of the difficulty of differentiating discrete curves the tangents are not actually computed. Instead a chord of a defined length is moved along the contour. At every point, the distance between the center of the chord and the curve is computed as a measure for the curvature. Figure 10.7 demonstrates this method for convex and concave portions of a contour. There is no difference in the computation, except that in one case the contour is located to the left of the chord connecting P_0 and P_1 , in the other case to the right. Both cases can be distinguished by the sign of the result. The definition of the sign is somewhat arbitrary, depending on the direction of the contour. One can arbitrarily define concave portions to have a negative curvature. Convex portions are then indicated by a positive curvature value.

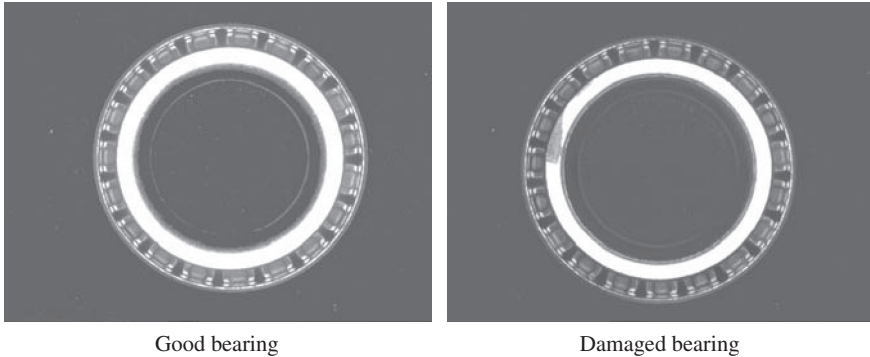


Fig. 10.8 Roller bearing inspection with curvature values

It is clear that the absolute value of this distance depends on the length of the chord. Because of the limit value in Eq. (10.8), this measure will be closer to the true value of the curvature the shorter the chord is. A shorter chord on the other hand will again lead to stronger variations due to the inevitable discretization of the contour in a digital image. In practice, it will rarely be required to actually measure the true curvature value. Rather, one will compare the curvature of the contour of a test piece with reference values computed under identical conditions.

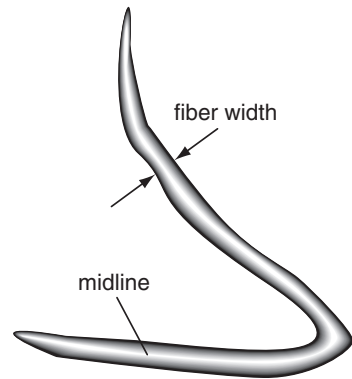
A typical application of the curvature is demonstrated in Fig. 10.8. It shows a top view of a common roller bearing. The closing ring of the inner shell of one of the bearings has been damaged by faulty machining. It is quite simple to segment the bright, reflecting portion of the ring as a contiguous object. The diameter of this ring is approx. 440 pixels. A curvature calculation with a chord of 40 pixels for a proper bearing results in a mean curvature of 0.571 with a standard deviation of 0.547. In contrast, the damaged bearing has a standard deviation of 1.206. The difference in maximum values is even more striking. The undamaged bearing has a minimum curvature of -1.27 and a maximum curvature 1.85, whereas the curvature of the damaged ring ranges from -7.7 to 7.13 (negative values indicating concave sections).

Curvature signature: In analogy to the distance-versus-angle signature from Sect. 3.2.2 the curvature values along the contour can also be collected in a vector. Like the distance signature, this curvature signature has a characteristic profile for different object shapes and can therefore be used for identification purposes.

10.2.2 Fiber Features

By the *fiber length* of an object we will understand the length of a line which has the maximum possible distance from the object contour at each point. For an object like the one in Fig. 10.9 this is obviously the length of the center line of the

Fig. 10.9 Fiber length and width of an object



object. The *fiber width* is then the distance of opposite contour points perpendicular to the center line.

The length of the center fiber can be computed according to Russ (2007) using a Euclidean distance map. All points inside the object are shaded according to their distance from the contour. Boundary points are set to black, the points at the greatest distance from the contour are set to white. The result of this process is shown in Fig. 10.9. It is of course quite computation-intensive to calculate the true Euclidean distance from the contour for each point of an object, but there is an effective iterative procedure to obtain the distance map:

1. Each pixel outside the object is first set to brightness 0, each point on the contour to brightness 1.
2. Each pixel inside the object adjacent to a point with brightness g is set to brightness $g + 1$. This procedure continues until every point has been assigned a brightness value.

This procedure creates an image analogous to that in Fig. 10.9 where the center line is brightly emphasized. The length of such a line can be computed according to Russ (2007) with a mean error of only 2.5 % according to the following formula:

$$L = 0.948 \times \text{Number of orthogonal neighbors} + 1.340 \times \text{Number of diagonal neighbors.} \quad (10.10)$$

For lines this formula is more precise than that used for computation of the perimeter of closed contours in Sect. 10.1.2.

10.2.3 Euler Number

The Euler number is based on the number of “holes” contained in an object. For a dark object segmented by binary thresholding this would be the number of bright

objects within its boundaries. Despite its simple definition, it is a very useful feature in technical applications. Many work pieces can be distinguished by the number of holes, e.g. different types of fasteners, car wheels or pipe flanges.

10.2.4 Moments and Fourier Descriptors

Moments: Moments can be used to describe objects with invariance to translation, rotation and scale. Moments of order $p + q$ are defined according to Nischwitz et al. (2007) by

$$m_{pq} = \sum_x \sum_y x^p y^q g(x, y). \quad (10.11)$$

with $g(x, y)$ the gray level of the point (x, y) . The sums run over all points belonging to the object. This definition is incidentally exactly equivalent to the one used in descriptive statistics. The same holds for the central moments defined using the object's centroid coordinates (x_c, y_c) :

$$\mu_{pq} = \sum_x \sum_y (x - x_c)^p (y - y_c)^q g(x, y). \quad (10.12)$$

In addition, these moments can be normalized:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}+1}}. \quad (10.13)$$

Using these normalized, centered moments, invariant features can be computed according to Hu (1962). For this, moments up to the third order are needed. Invariant moments are a very powerful method for object description, but, as can be seen from the formulas, it is very computation intensive. It is therefore rarely used in industrial applications. The mathematical properties of moments are discussed in more detail in Sonka et al. (2008), who also provide extensive references.

Fourier descriptors: Another elaborate but powerful method for object description is based on the Fourier transform of contour representations. Gonzalez and Woods (2008) treat every point of the contour as a complex number with the x -coordinate as the real part and the y -coordinate as the imaginary part. The whole contour is then represented as a sequence of complex numbers. The Fourier coefficients of this sequence can be used to describe the contour. Sonka et al. (2008) list several methods for the computation of invariant features from Fourier descriptors. This method of object description is often used for handwritten character recognition.

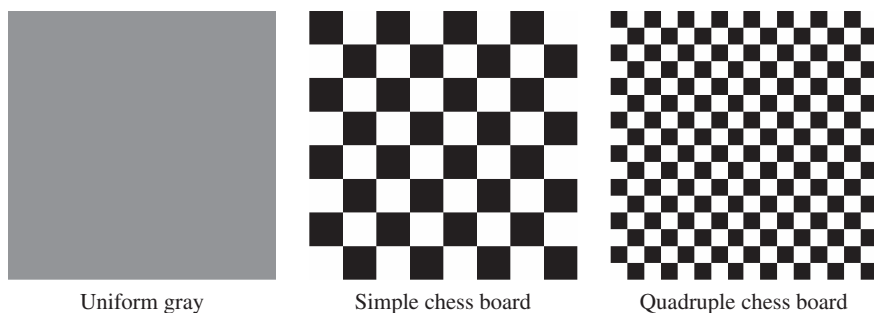


Fig. 10.10 Objects with the same mean gray level

10.3 Gray Level Features

Gray level features describe the brightness distribution within an object. This group contains some of the most basic features as well as very complex description methods for surface textures.

10.3.1 First-Order Statistics

Formulas for the computation of first-order statistics contain only the gray level of a single pixel—hence the name. Typical first-order features are:

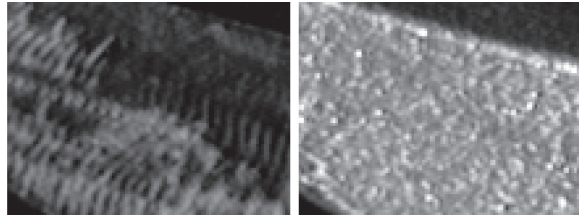
- Average gray level,
- Minimum gray level,
- Maximum gray level,
- Standard deviation of gray levels.

Obviously, objects of very different appearance can have the same first-order statistics. Figure 10.10 shows three different shapes with the same mean gray level of 127. In all other features of first-order gray level statistics, the homogeneous gray image differs from the two checkered images. For example, the simple checkerboard has a standard deviation of 127 while the homogeneous gray area has a standard deviation of 0. However, it is impossible to tell the two checkered images of different sizes apart. The reason for this is that the spatial relation between pixels is not taken into account.

10.3.2 Textural Features

Texture denotes the appearance of surface structures, e.g. the typical finish of steel, granite, aggregate concrete, etc. Texture is one of the concepts in image processing

Fig. 10.11 Polished and unpolished aluminum surface



Polished, average gradient 15.3

Raw, average gradient 25.1

most difficult to define precisely. Consequently, there are many different methods to describe textures quantitatively. Many of these methods are based on statistical approaches. It has been shown in Julesz (1975) that humans cannot distinguish between textures with identical second-order statistics. Second-order statistics evaluate relationships between pairs of pixels. The pixel pairs are defined by essentially arbitrary displacement vectors.

The most basic second-order features are based on gradients, i.e. gray level transitions between neighboring pixels. The two chess board patterns in Fig. 10.10 can easily be distinguished using gradient features. The small checkerboard pattern contains precisely the same number of black and white pixels, but there are twice as many transitions between the two colors. Correspondingly, the mean gradient is twice as high. These simple features are already sufficient to differentiate between many technical surfaces and to detect defects on these surfaces. Figure 10.11 shows a comparison between a polished and an unpolished aluminum surface under a very flat illumination angle. The two surfaces can easily be distinguished by their mean gradient.

Texture analysis is a large and varied subject. Depending on the surfaces to be inspected, it may require considerable mathematical effort, up to the simulation of specific surfaces using specially designed stochastic models. The variety of surface analysis methods has led to solutions being developed in the area of texture analysis that are specifically tailored for each application; this makes it impossible to give typical solutions based on established methods. For further information please refer to literature, e.g. Davis (1981), van Gool et al. (1983), Waszkewitz (1999), Tönnies (2005), Jähne (2005), Nischwitz et al. (2007), Sonka et al. (2008), Gonzalez and Woods (2008).

10.4 Summary

Object features: Objects segmented from the image scene can be characterized by various features in order to recognize, classify and inspect them. Most geometrical features are only meaningful for objects found by thresholding or edge-based methods because objects segmented by template matching do not necessarily correspond to visually contiguous image structures.

Enclosing rectangle: Coordinates and dimensions of the enclosing rectangle are some of the most basic geometric object features. The rectangle can be aligned with the coordinate axes or, for a more precise description, along the principal axis of the object.

Rectangularity: This feature is defined as the ratio between the area of the object and the area of the enclosing rectangle, which should always be aligned with the object's principal axis to compute this feature precisely.

Area: The area A of an object is defined as the number of image pixels within the object boundaries.

Perimeter: The perimeter P of an object can be determined directly from the contour description in chain code. Diagonally adjacent pixels contribute to the perimeter with a length of $\sqrt{2}$, horizontally or vertically adjacent ones with a length of 1. The perimeter is very sensitive to noise and image resolution, but it can be used successfully for the description of the typically rather smooth object edges prevailing in industrial scenes.

Form factor: This feature is defined as $4\pi A/P^2$. For an ideal circle it has its maximum value of 1. For any other object it is smaller.

Centroid: The location of the object's centroid can be computed from all pixels of the object or—using appropriate correction terms—directly from the contour. The coordinates of the centroid are subpixel accurate due to the averaging over the entire object.

Principal axis of inertia: The principal axis of an object is usually determined by approximate methods because of the computational effort required for a formal calculation. Common methods assume, for example, that the principal axis passes directly through contour points or use an enclosing polygon to approximate the contour and compute the axis from the polygon.

Maximum Feret's diameter: Feret's diameter is the length of the projection of an object to an arbitrarily oriented axis. The maximum Feret's diameter is the maximum distance between two points on the object's contour.

Radii: An object radius denotes the distance between a contour point and the object's centroid. The minimum and maximum radii, along with their relative orientation, can be used to describe the shape of an object.

Contour curvature: Formally, the curvature is computed from the second derivatives of the contour. Because of the noise sensitivity of numerical differentiation, the distance of contour points from a fixed-length chord is commonly used as a measure for the curvature. Convex and concave portions of the contour can be distinguished using the sign of the curvature value. Curvature values can, for example, be used to detect damages to the boundaries of an object.

Curvature signature: The curvature signature contains the curvature values for all points of the contour. Because of its characteristic profile it can be used for object recognition.

Fiber length and width: The center line or fiber of an object is a line within the object having the largest possible distance from all points of the contour. It can be

found using an Euclidean distance map. Its length is called fiber length, the average distance of contour points from this line is defined as the fiber width.

Euler number: This feature gives the number of “holes” in an object. For a dark object found by binary thresholding, this would be the number of white areas within the object. Many technical objects, e.g. punched parts, can be distinguished by their Euler number.

Moments: These features are defined in analogy to the moments of descriptive statistics. They can be used to describe objects with invariance to translation, rotation and scale, but they are very computation-intensive.

Fourier descriptors: Denote the coefficients of a Fourier transform of an object contour. These features are also very computation-intensive and are used mainly in handwritten character recognition.

First-order gray level statistics: First-order statistical features use only the gray levels of individual pixels. They are not able to describe spatial relationships. An example is the average brightness of an object.

Second-order gray level statistics: Second-order gray level statistics use the gray levels of pixel pairs with defined displacement vectors. With directly neighboring pixels this is called gradient statistics. This can be used to describe simple textures.

Textural features: A multitude of features has been developed for the description of textures. Most use second-order gray level statistics, sometimes based on highly complex stochastic models.

References

- Bronstein IN, Semendjajew KA, Musiol G, Mühlig H (2005) Taschenbuch der Mathematik, 6th edn. Harri Deutsch, Frankfurt a. M
- Davis LS (1981) Image texture analysis techniques—a survey. In: Simon JC, Haralick RM (eds) Digital image processing. Reidel, Boston
- Gonzalez RC, Woods RE (2008) Digital image processing, 3rd edn. Pearson Education, Upper Saddle River
- Hu MK (1962) Visual pattern recognition by moment invariants. IRE Trans Inf Theory 8(2):179–187
- Jähne B (2005) Digitale Bildverarbeitung, 6th edn. Springer, Berlin
- Julesz B (1975) Experiments in the visual perception of texture. Sci Am 232(4):34–43
- Mandelbrot B (1987) Fractal geometry of nature. W.H. Freeman & Co., New York
- Nischwitz A, Fischer M, Haberäcker P (2007) Computergrafik und Bildverarbeitung, 2nd edn. Vieweg, Wiesbaden
- Parker JR (1994) Practical computer vision using C. Wiley, New York
- Russ JC (2007) The image processing handbook, 5th edn. CRC Press, Boca Raton
- Sonka M, Hlavac V, Roger B (2008) Image processing, analysis, and machine vision, 3rd edn. Cengage Learning, Stamford
- Tönnies KD (2005) Grundlagen der Bildverarbeitung. Pearson Studium, Boston
- van Gool L, Dewaele P, Oosterlinck A (1983) Texture analysis anno 1983. Comput Vision Graph Image Proc 29(3):336–357
- Waszkewitz P (1999) Detektierung von Beschriftungen auf metallischen Oberflächen mit Hilfe von Texturmethode und Neuronalen Netzen. PhD thesis, University of Stuttgart

Chapter 11

Color Image Processing

This chapter will present an introduction to color image processing. We define color image processing as the evaluation of color information in images for industrial image processing applications. Even though color television has been the standard for decades and the consumer market offers almost exclusively color cameras, most inspection lines in machine vision are still equipped with gray level cameras, and inspection tasks are solved evaluating only the brightness information. This is even more remarkable considering that color is an essential part of the visual information perceived by humans. According to Russ (2007), a human can distinguish merely 20–30 gray levels but about 1,000 colors and shades of colors; this illustrates how important color information is for us.

There are, however, several drawbacks to using color images in industrial image processing. First of all, color images contain more information about the scene captured than gray level images, and this additional information entails an additional need for memory space and computational effort. Additional effort that one will want to avoid if possible. Another reason is that most image processing algorithms were developed for gray level images and have been proved for the evaluation of gray level images. Transferring these established methods to color images is not always easily possible. Furthermore, describing color is far more complex than describing brightness. While only a single value is necessary to describe the brightness of one pixel, three mutually independent values are usually needed to describe color while the meaning of the three individual values is not fixed. As we will see in our first application example, there are various methods with which to describe color quantitatively, each with its advantages and disadvantages.

Yet it is necessary for some applications to use color image processing. There are two typical questions. One is determining the current color of an already detected object. This is an identification task taking the color of the object as characteristic for classification. We will encounter this sort of task in the first application example.

Another typical task is verifying the position or presence of an object of a specific color. This is a segmentation based on color information where the object contrasts with the background because of its color. After segmentation the objects

can be used just as well for position and presence verification, and gauging as segmented objects from gray level images. An example for color segmentation will be presented in the second application of this chapter.

11.1 Color Identification

Our first example for color image processing is from the area of textile retail. We probably all know the little colorful plastic rings on clothes hangers in department stores and clothes stores indicating the size of the garment. When the article of clothing is sold, the size ring is removed. The large number of size rings obtained this way are completely mixed as can be seen in Fig. 11.1. To recycle the size rings, they have to be sorted according to indicated garment size, which is the task of this application.



Fig. 11.1 Size rings before sorting (courtesy of Industrial Vision Systems Ltd., Oxford, UK)

11.1.1 Evaluation Strategy

The decisive criterion for sorting is the white number or letter combination stamped on the size ring.¹ We could, of course, solve this task with character recognition in a gray level image as described in Chap. 5; however, there are some objections to this approach.

- Since the objects are round, there are always perspective distortions of the numbers as can be seen in Fig. 11.2 which makes the classification less robust. This can be problematic since about 70 different sizes have to be identified.
- Feeding of the size rings ensures that only one object at a time is within the capturing area of the camera; however, the rings' orientation is random so the writing on half of the size rings is upside down as can also be seen in Fig. 11.2. Thus, character recognition has to be able to recognize upside-down characters—almost doubling the number of classes—or the numbers are read twice and the image is turned upside down between reading, but the two readings also require twice as much computation time.
- If template matching is used for identifying the sizes, a large number of sizes entails high computational effort which can pose a problem with the required cycle time of 300 size rings per minute.

Because of the reasons mentioned above, the obvious approach to identify the size using character recognition is ruled out. The second preeminent characteristic of the size rings is their color and this lends itself to using color as a means of identification. The various sizes have different colors so to better tell the sizes apart when they are still on the rack. Actually, the color of the size rings is not unambiguous. Each size has only one color; however, one color is used for various sizes. Most of the time the sizes with the same color are not on the same rack since

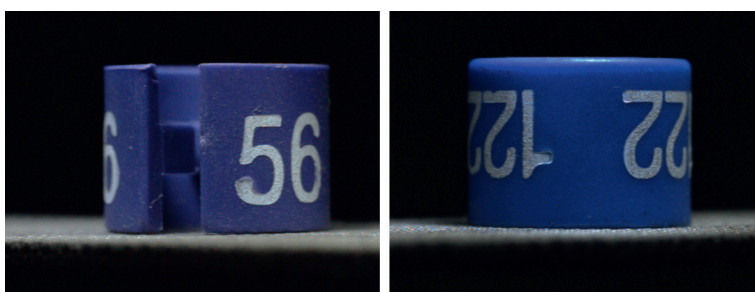


Fig. 11.2 Perspective distortions because of round object bodies

¹ Since all sizes except for S, M, L, and so on are labeled with numbers, we will only speak of numbers in the following, which will also include the sizes denoted by letters.

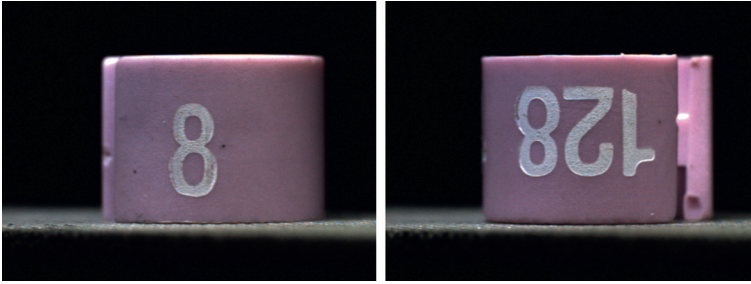


Fig. 11.3 Different sizes on rings of identical color

one is the British women size 8 (Fig. 11.3 left) and the other the children size 128 (Fig. 11.3 right). For our identification task, this means that the size rings' color does not suffice as the only criterion.

Actually it is a good idea to combine both possible criteria. In the first step, identifying the color or color class can achieve a rough classification of the size rings. In the second step, the character recognition merely has to differentiate between a few sizes. By combining the two identification methods, it is possible to sort the size rings with sufficient reliability and at a high speed.

11.1.2 Illumination and Image Capture

The size rings are transported through the inspection station lying on a conveyor belt. To be able to capture the writing as well as possible, the camera should be pointed horizontally at the ring. Since each number is printed three times on each size ring, the camera captures the image at an angle of 60° in the worst case. As you can see in Fig. 11.4 left, the writing can hardly be identified from a 60° -angle. Therefore, the size rings are captured by a second camera whose line of view is turned by 60° from the first camera as sketched in Fig. 11.5. As can be seen from the right side of Fig. 11.4, the camera will thus be pointed directly at the writing

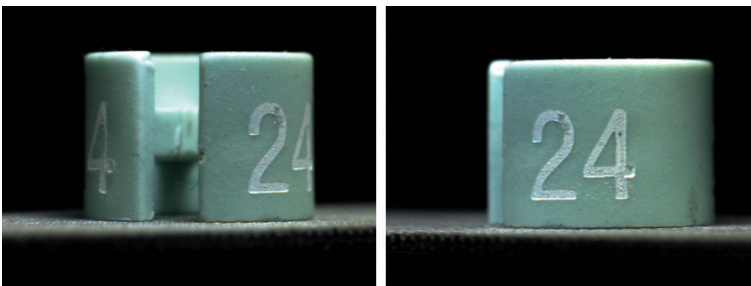


Fig. 11.4 Size ring as seen by two cameras offset by 60°

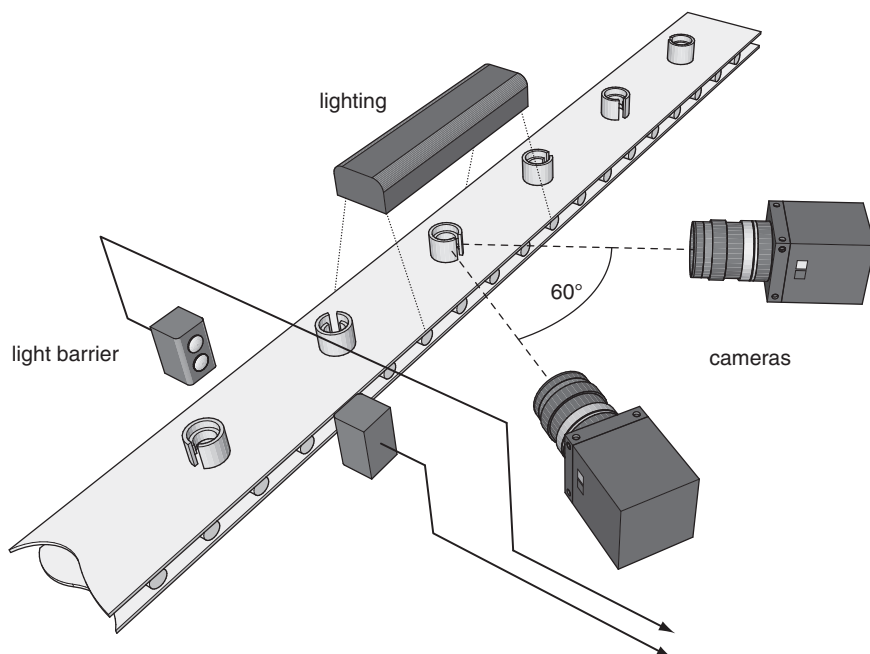


Fig. 11.5 Station setup for recognizing size rings

while the other camera is looking between two writings. With this configuration using two cameras, in the worst case the writing is turned by 30° with respect to the camera's axis which does not pose a problem for identification.

Image capture is triggered by a light barrier so the position of the object in the field of view of the camera is quite constant. Since the writing is quite large compared to the object, color cameras with a very low resolution of 640×480 pixels are quite sufficient.

To enable color recognition, front lighting is indispensable. However, horizontal lighting is ruled out because of the cylindrical shape of the size rings, the light would always be reflected directly into the camera disturbing both character and color recognition. That leaves only lighting from above. This should be rather diffuse to enable color recognition over the entire ring width and is done using a line-shaped lighting element made up of white-light LEDs. Fortunately, the contrast between white writing and colored ring body is good enough for all colors to allow character recognition with diffuse front lighting. If a dark field were necessary for character recognition, it would hardly be possible to use the same camera and illumination to capture the color of the ring. The intensive light of the LEDs creates a bright halo at the upper edge of the rings with some colors; however, this does not disturb character recognition since the halo is far enough away from the writing, and during color recognition, this area is exempted from evaluation.

11.1.3 Color Classification

The first step in evaluation is categorizing the size rings into color classes. For this, a color description is necessary enabling the computer to do the classification. We probably all know of cases where two people call the same color by different names; for example, one may call an object yellow, which is called orange by another. For classification, a quantitative description of the color is therefore necessary.

The easiest way to describe color is the use of RGB values as a basis for classification. The RGB color space is very well suited to describe color data captured by a camera. However, there are two reasons against using the RGB color space: for one, the depiction is unlike the human perception of colors which can complicate the interpretation of values, and the evaluation of color cannot be separated from brightness. Usually, a human separates the color from brightness, for example light blue and dark blue is still the same color to us with different levels of brightness; while a light blue and a light green have the same brightness but different color hues. However, color hue and brightness are not sufficient to describe a color completely: It is also necessary to be able to tell apart a vivid from a pastel color. This color property is called saturation; intense, “vivid” colors have high saturation whereas pale or “washed-out” colors have low saturation.

The HSI color space describes colors in this way; the name is derived from hue, saturation, and intensity. With hue, the HSI color space draws on the traditional image of the color wheel describing the hue with an angle. Red is at zero degrees; the angle then rises over yellow, green, blue and violet to return to red at 360°. The color wheel turns into a color disc with introducing color saturation as radius. In the center, i.e. a saturation of zero, the color disc is gray, while on the edge of the color disc the completely saturated colors are located. Color discs for various levels of brightness can be seen in Fig. 11.6. Color hue as angle, saturation as radius and intensity as height make up a cylinder coordinate system describing the color space. As can be seen from Fig. 11.6, the color disc is the largest for medium brightness. Actually, colors can only be fully saturated at medium brightness. In the extreme cases of maximum and minimum brightness saturation is not even possible since white and black have no pigment content.

Since the image data of most cameras is from the RGB color space, the color data has to be transformed into the HSI color space. According to Gonzalez and Woods (2008) using

$$\begin{aligned}
 H &= \begin{cases} \theta & , \text{ if } B \leq G \\ 360^\circ - \theta & , \text{ if } G < B \end{cases} \\
 S &= 1 - \frac{\text{Min}\{R, G, B\}}{R + G + B} \quad \text{and} \\
 I &= \frac{1}{3}(R + G + B)
 \end{aligned} \tag{11.1}$$

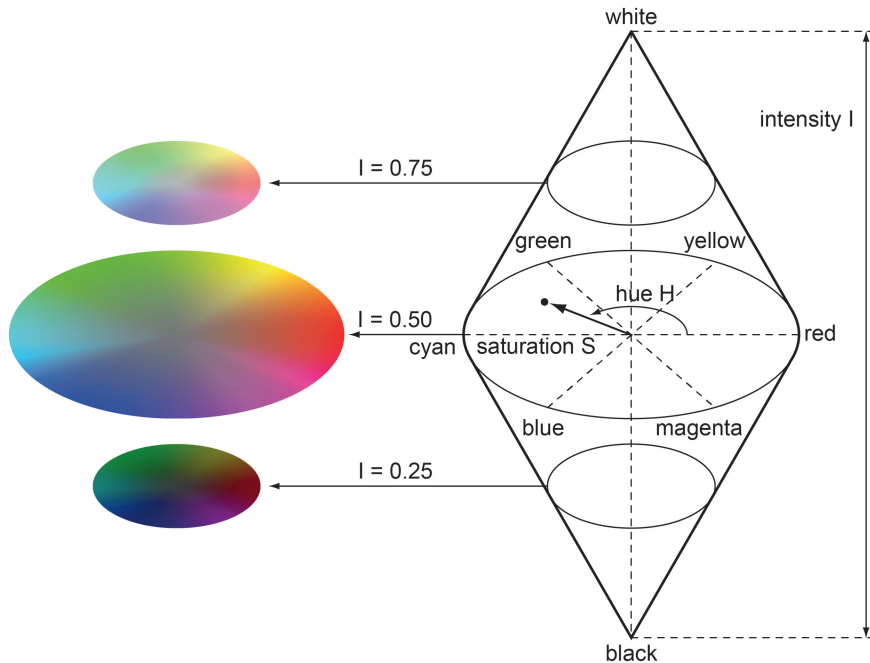


Fig. 11.6 The HSI color space

RGB color data can be transformed into the HSI color space with

$$\theta = \arccos \left(\frac{2R - G - B}{2\sqrt{(R - G)^2 + (R - B)(G - B)}} \right).$$

Having introduced the HSI color space, we can now turn to the inspection itself. First, we have to define the area in which to determine the color. Of course, the color could be determined using the entire object, but then the white writing would also be part of the evaluation. Therefore, it is better to use only a partial area to determine the color. Because of the illumination, there can be reflections in the area above the writing falsifying the color determination, which is why the area below the writing is best. Because of the round shape of the object, the edges of the size ring appear darker than the center of the ring. Although the description of the color is independent from brightness in the HSI color space, colors cannot be recognized in very dark areas so there has to be some space to the left and right of the search area. That leaves a stretch below the writing as evaluation area.

For this application it is unnecessary to position the ROI. Because of the light barrier triggering image capture and the constant speed of the conveyor belt, the horizontal position of the object is sufficiently stable. In vertical direction the

position is determined by the conveyor belt. However, it has to be taken into account that the distances between the writing and the top and bottom edges of the size ring are different. Therefore, the height of the bottom edge of the writing depends on the ring being upside down or not.

The color determination itself consists in converting the image into the HSI color space and then to determine the mean gray level in the search area for each channel (H, S and I). To reduce the effects of camera noise and staining of the size ring, the measurement area should be smoothed before being transformed into the HSI color space.

For each size ring color the acceptable range for the three color coordinates must be stored. With 70 possible colors, this means in the worst case that the measured values have to be compared to the given values 70 times. To reduce the number of comparisons, we recommend pre-classification. This categorizes the colors according to hue into eight classes segmenting the color disc into 45°-areas. Depending on the area, only five to twenty comparisons are needed for color classification.

One problem with color classification remains: the rings have a gap to stick them on the hangers. These areas are not illuminated well so they appear much darker than the rest of the size ring's surface. These areas have to be exempted from color evaluation; so instead of one large evaluation area, we will use several small ones as depicted in Fig. 11.7. Thus the evaluation areas located in the region of the ring gap can be removed before the color determination. The evaluation areas that are completely in the shadow can be recognized by their low brightness, while the transitions between shaded areas and colored surface is recognizable by the high brightness contrast. Before determining the color, those evaluation areas are sorted out that are either very dark or show a high contrast. Thus the shadowed areas of the gap are not part of the color evaluation.

Using several evaluation areas for color recognition also allows for suppressing outliers by eliminating the evaluation areas with the largest and smallest color values from evaluation. Color evaluation is summarized in Table 11.1.

Fig. 11.7 Regions for determining the color

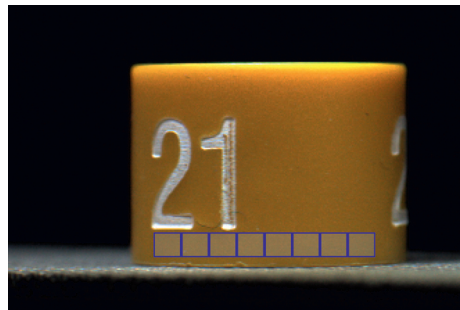


Table 11.1 The following table summarizes the example from Sect. 11.1.3

Category	Description
Directory	\\Examples\\Color
Illumination	Front lighting with white light LED
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i> , defines several small ROIs below the writing with sufficient distance to the side edges 3. <i>Convert color image</i> , converts the image into an intensity image 4. <i>Compute features</i> , determines brightness and contrast 5. <i>Screen ROIs</i> , eliminates the gap of the size ring from color evaluation using brightness and contrast 6. <i>Convert color image</i> , converts the image into a hue image 7. <i>Copy ROIs</i> , copies the ROIs for determining the hue 8. <i>Compute features</i> , determines the hue as mean gray level 9. <i>Sort ROIs</i> , removes the ROI with the highest hue to suppress outliers 10. <i>Sort ROIs</i> , removes the ROI with the lowest hue to suppress outliers 11. <i>Convert color image</i> , converts the image into a saturation image 12. <i>Compute features</i> , determines the saturation as mean gray level 13. <i>Sort ROIs</i> , removes the ROI with the highest saturation to suppress outliers 14. <i>Sort ROIs</i> , removes the ROI with the lowest saturation to suppress outliers 15. <i>Combine ROIs</i> , combines the ROIs of hue and saturation values

11.1.4 Selecting a Camera Image for Character Recognition

As mentioned above, identifying the color of the size ring is merely the first step of inspection. The second step is character recognition. Two cameras are used to ensure that the writing is at least legible in one camera image. Since it is not clear which one to use, the next step is to determine the camera image where the writing is more legible. The writing is the better to read, the straighter the camera points to it. If the camera is pointed directly at the writing, the characters are much wider than if being captured under a large angle. This fact is utilized by choosing the image for character recognition where the width of the writing is largest. The next step is thus to measure the width of the writing in both camera images and select the one with the greater width for reading the size. To measure the width of the writing, the writing must be segmented as an object, which for a size value consisting of several numbers means to combine the numbers.

Since the writing is always white, it is always brighter than the ring color. Therefore, measuring the width of the writing can be done using a gray level image representing the intensity of the color image. To get this intensity image, we could transform the image into the HSI space and then use only the intensity channel. However, this kind of effort is usually avoided and the intensity image is computed directly from the RGB image by computing a mean value from the red, green and blue value. When adding all three color values with the same weight, the resulting intensity image does not correspond to a human’s sense of intensity. Actually,

green light influences the perception of intensity by a human much more than red or blue light. Therefore, each of the color channels is weighted before contributing to the intensity value. Russ (2007) gives the following weights for transformation:

$$H(x, y) = 0.299 \cdot R(x, y) + 0.587 \cdot G(x, y) + 0.114 \cdot B(x, y) \quad (11.2)$$

Using this formula, an intensity value can be calculated directly for each pixel of the color image, and thus the color image can be transformed into a gray level image. We can see from Eq. (11.2) that the green value contributes more than half to the intensity value.

Because of the various levels of brightness of the background, we recommend segmenting the writing using the edges to the background. For this we use a contrast filter with a large filter kernel whose result can be seen in Fig. 11.8 left. The edges of the writing now appear bright in the image. But the edges of the ring appear bright, too, since the ring contrasts more or less with the background depending on color. Contrast between writing and ring also obviously depends on the ring color. To balance these differences, the image is normalized as depicted in Fig. 11.8 right.

After contrast filtering and normalizing, the edges of the numbers are completely bright in the image while the inside of the numbers and the rest of the size ring are dark. In this edge image the numbers are thresholded; the ROI can be selected to exclude the top and bottom edges of the size ring. When thresholding, please remember to have the found objects filled since otherwise the dark insides of the numbers would remain as holes in the object. Apart from the desired numbers, the left and right edges of the ring are also created as objects. Since this edge is higher and thinner than the numbers, it can be removed by filtering using these features.

To combine the digits of one number, we create an artificial image where the found objects are depicted in white and everything else is black, as can be seen in Fig. 11.9 left. Dilating the digits several times with a specially adapted filter kernel makes them wider so the digits of a number form a continuous white area as can be seen in Fig. 11.9 right. A horizontal line—one pixel high and eleven pixels wide—is used as filter kernel. Dilating with this mask makes the objects wider but their height remains unchanged. For the size numbers whose digits are relatively close to each other, this causes the digits to fuse into one bright area.

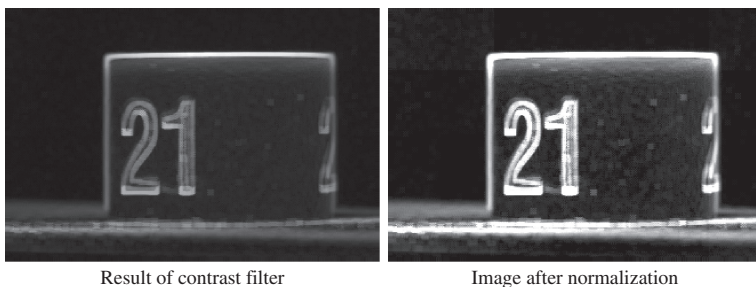


Fig. 11.8 Image after contrast filtering and normalizing

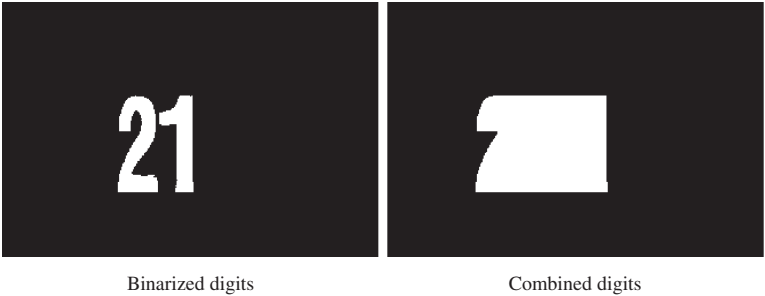


Fig. 11.9 Drawn and combined objects

The digits thus fused can then be segmented as a single object using thresholding. With an adverse angle of vision of the camera, two writings may be visible in the image. In this case the same applies to the two writings in one image as for the writings in two different camera images. The wider writing is better suited for recognizing the size. Therefore, only the widest object is used for analysis and the other one discarded.

The width of the remaining writing can then be measured and the camera image with the wider writing selected; this is the image on the right in Fig. 11.10. Because of dilation the measured width is significantly larger than the actual width as can be seen in Fig. 11.10. The procedure described here cannot be used for gauging the actual width of the writing. To select the proper camera image, only a comparison between the two widths is necessary. Determining the width of the writing for selecting the better camera image is summarized in Table 11.2.

11.1.5 Recognition of Writing

To recognize the writing, one can either sample the characters and then identify them using a classifier, or search the characters directly using template matching.

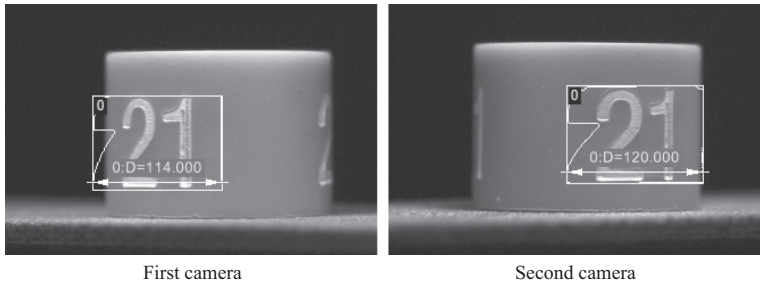


Fig. 11.10 Widths of the dilated writing in both camera images

Table 11.2 The following table summarizes the example from Sect. 11.1.4.

Category	Description
Directory	\Examples\Color
Illumination	Front lighting with white light LED
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Convert color image</i>, converts the image into an intensity image 3. <i>Resize image</i>, halves the image size to allow for a faster evaluation 4. <i>Filter image</i>, enhances the gray level transitions by filtering using a contrast filter 5. <i>Normalize image</i>, enhances image contrast 6. <i>Define ROIs</i>, defines a search area covering the entire image width at writing level 7. <i>Determine threshold</i>, computes a threshold for segmenting the writing 8. <i>Create ROIs by thresholding</i>, segments the numbers 9. <i>Compute Features</i>, computes height, width and area of the objects to sort out the edge on the side of the size ring 10. <i>Screen ROIs</i>, filtering removes the edge on the side of the size ring and other spurious elements 11. <i>Draw ROIs</i>, draws an artificial image where the numbers found are white and the remaining image is black 12. <i>Filter image in ROIs</i>, dilates the artificial digits to fuse them together 13. <i>Create ROIs by thresholding</i>, creates an object from the merged digits 14. <i>Compute features</i>, computes the width of the objects to determine the widest group of digits in the image 15. <i>Sort ROIs</i>, sorts out all objects except the widest 16. <i>Gauge ROIs</i>, determines the width of the best legible writing in the image 17. <i>Copy measurement list to tray</i>, copies the width of the writing to the tray to compare it to the width of the other camera image

Measuring is done for both camera images

With this application we decided to use template matching. We did not use a classifier because sampling the characters is very sensitive to distortions because of the object’s round shape. Another reason was the large amount of training data that had to be created because of the many different clothes sizes. On average, template matching requires only three to four samples for one clothes size. To train a neural net, however, many more samples are necessary resulting in more effort. Finally, the significant difference in width between the clothes sizes with one digit and those with three makes sampling the characters rather difficult. Of course, we could have read every digit individually; however, this would have complicated segmentation.

For this application, the run time of template matching could have been problematic. However, the number of templates that are searched for is low enough to allow a short cycle time. There are two reasons for the low number of templates: first of all, as mentioned before it suffices to search for three or four templates per clothes size. Second, because of the preceding color classification, the number of possible sizes is reduced to two to three, and is thus very low. On average, ten templates need to be searched for per color class.

Two more aspects have a positive effect on the runtime of template matching: To determine which camera captures the better image of the clothes size, we created the writing by thresholding as in Sect. 11.1.4. This object is very well suited as position reference to position the search area for template matching. Thus the search area for template matching can be chosen rather small. Also, the writing is a large-area object which allows for a large step size for rough search. With this application we were even able to go a step further: To reduce computational effort, the image size was divided in half even before determining the better camera perspective, i.e. every other column and row of the image was deleted. The remaining pixels suffice both for selecting the image and recognizing the writing using template matching. Because of these conditions, it is possible to execute the inspection with template matching fast enough for this application.

We already mentioned that the size rings can be upside down. Upside down writing is obviously not recognized by template matching. There are different ways of dealing with this problem:

- We can either store upside down templates leading to more templates.
- The templates can be rotated during runtime which leads to additional computational effort per template.
- We can rotate the image and search again, which causes only one computation per image.

With this application the image is rotated to limit the number of templates and because rotating the image takes less time than rotating ten templates.

It is possible to do template matching with the color image and compute the correlation of the color values. However, in this application we have already used the color information for categorizing the rings. Thus template matching is only done for clothes sizes with the same or very similar colors so the correlation in the color image has no more additional information than the intensity image. Also, computing the correlation in the color image requires three times more computational effort than correlation in the gray level image. Because of these reasons the intensity image we used for determining the width of the writing is also used for recognizing the clothes size. Table 11.3 summarizes the identification of the writing.

Table 11.3 The following table summarizes the example from Sect. 11.1.5

Category	Description
Directory	\\Examples\\Color
Illumination	Front lighting with white light LED
Check functions	The example uses the following check function sequence: 1. <i>Copy image from tray</i>, the reduced image used for determining the writing width is used here 2. <i>Define ROIs</i>, sets a search area for character recognition. The position is taken from determining the writing width 3. <i>Template matching</i>, looks for the clothes sizes associated with this color 4. <i>Count ROIs</i>, checks whether a proper clothes size was found

If no clothes size is found, the image is rotated and the search restarted

11.2 Color Segmentation

This example is taken from the electrical industry. It is a relay where the galvanization of the ends of the conductor wire has to be inspected. The wire is made of coated copper. To make a conductive connection with the contact, the end is galvanized. During this process, the isolating coating evaporates so the surface of the wire end is conductive. Figure 11.11 shows one coil of the copper conductor and the galvanized wire end clamped on a brass contact.

This inspection task is literally a presence verification because we need to check whether the galvanized end of a wire is on the contact or not. The tricky part of this inspection task is that there is not enough difference in the brightness of the not galvanized and the galvanized wire and the contact for stable thresholding. However, the three metals are quite distinct in their color and thus the task can be solved by color segmentation.

11.2.1 Illumination

Metal surfaces reflect light strongly. Therefore, they are rather difficult to illuminate especially when they are curved as wires are. The incident light is almost always reflected directly into the camera from some point. One can try to minimize the reflections with very flat dark field illumination. In this case, however, this is not possible since the edge of the relay protrudes the contact. So with flat light the contact is in the shade.

Since also the reflected light has the typical color characteristic of each metal, color segmentation can be done using the reflected light. For this the exposure time of the camera must be appropriately short or a sufficiently large f-number must be used so the image is not overexposed. A ring light with white light LEDs from the direction of the camera is used for illumination.

Fig. 11.11 Galvanized end of a copper wire on brass contact



11.2.2 Color Classification

The segmentation with thresholding presented in Sect. 4.3.1 was the first step to categorize, i.e. classify, the pixels into white and black. In a second step, the segmentation proper, pixels of the same logical color were combined to objects. For the segmentation in color images the basic procedure is the same: In the first step, each individual pixel is categorized, i.e. classified, according to its color into a color class. In the second step, neighboring pixels of one color class are combined into objects as described in Sect. 11.2.3.

We have already described a color classification in the previous example. By averaging many pixels, the color can be determined very accurately and in a robust way, which is why the borders of the numerous color classes can be very narrow. In this application, however, conditions are different: here we have to decide for each pixel whether it belongs to one of the three metals, i.e. color classes, or is part of the background. The color values of individual pixels of one color class are clearly distinct, unlike the average color values of the ROIs from the previous example. The spread of the color values for the wire is caused in part by the curvature and thus the changing angle between illumination and surface; but also by the changing angle between the surface and camera on the edges. The rough texture of the contact plate also contributes to very different colors that are all part of brass. This requires a classification based upon freely definable class borders.

For this kind of classification task we introduced the Nearest Neighbor Classifier in Sect. 6.3. By selecting the reference colors—this corresponds to the prototype pattern in the nomenclature of Sect. 6.3—the borders between the classes can be described as accurately as necessary without an analytic description of the border existing. As feature vector the RGB values of the pixels can be used directly. Transformation into the HSI color space is not advantageous since the class borders are arbitrarily defined because of the selected reference colors, independent of the color space. Quite the contrary: The distance computation between two feature vectors necessary for the nearest neighbor classifier requires more effort in a cylindrical coordinate system like the HSI color space than in the Cartesian coordinate system of the RGB color space. The distance d_j between the color of a pixel i and reference p_j is calculated following Eq. (6.6):

$$d_j = \sqrt{(i_R - p_{jR})^2 + (i_G - p_{jG})^2 + (i_B - p_{jB})^2}. \quad (11.3)$$

The relevant colors—zinc, brass and copper—in this example only take up a small part of the entire color space. The large part remaining is the background that is mostly plastic from the casing, shadow etc. In Sect. 6.3 we have already pointed out that the classes protrude the area covered by the references leading to completely random results. Because of these issues, the nearest neighbor classifier was extended by a rejection threshold as described in Sect. 6.3.2. This extension is also used for color classification.

The determination of the color class of a pixel starts with computing the distance to the nearest reference color according to Eq. (11.3). If this distance is smaller than the rejection threshold, the pixel is allocated to the class the reference color belongs to. If the distance is larger than the rejection threshold, the pixel is classified as background.

The area of the color space belonging to one class is thus defined by the selected reference color and the rejection threshold. However, the rejection threshold is only relevant for the separation between the colors and the background. The separation of colors among each other is only determined by the reference colors. A low rejection threshold makes the border between the color and the background run very close to the reference points which allows for a very accurate and detailed border line. However, this requires a high density of reference points leading to many sample colors if a larger color area belongs to the class. As discussed in Sect. 6.3, the computation time for nearest neighbor classifiers rises with the number of prototypes which is why as few reference colors as possible should be used for a quick classification.

A high rejection threshold allows for a larger distance among the reference colors to be defined. Thus a larger area of the color space can be allocated to one color with fewer reference colors. On the other hand, the borders of the area can be influenced less strongly. The selection of reference colors and the setting of the rejection threshold is thus a compromise between a stable classification with area borders as exact as possible, and a fast classification with as few reference colors as possible. Stability does not only depend on the number of reference colors but also on the colors selected. If necessary, speed can be increased by optimizing the reference color selection without affecting stability, but this requires longer experimentation. Figure 11.12 shows segmentation results with eight reference colors for zinc, two for copper and eighteen for brass. The high number of reference colors for brass is necessary because of the textured surface. The left figure shows the smoothed image. In the middle a well suited rejection threshold for the reference color selection is used. In the right figure a higher rejection threshold is used with the same reference colors which is why many pixels that are classified as background with the lower threshold are wrongly segmented as objects.

In Fig. 11.13. the results of the segmentation with the same rejection thresholds are displayed as in Fig. 11.12. However, the number of reference colors was

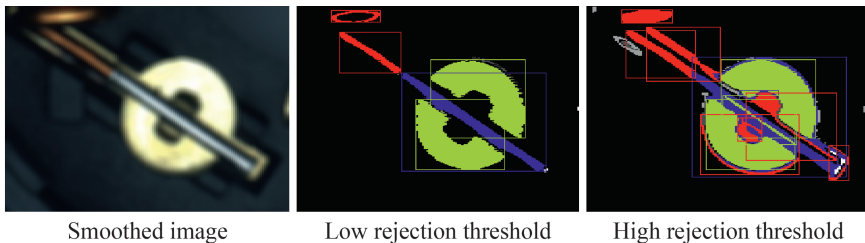


Fig. 11.12 Segmentation with various rejection thresholds

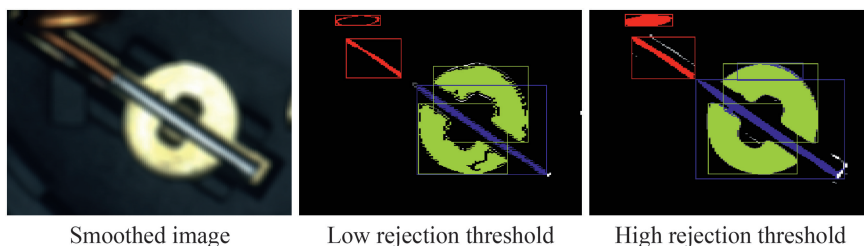


Fig. 11.13 Segmentation with few reference colors

significantly reduced to four for zinc, one for copper and eight for brass. As one can see in the middle figure, the lower threshold does not fit the reference data since large parts of the object are not segmented creating holes in the objects; on the other hand, the high threshold fits this selection very well as can be seen in the right figure. Compared with the previous figure, the number of reference colors was approximately cut in half; this also reduces computation time by half. As can be well seen in Figs. 11.12 and 11.13, the reference color selection and the rejection threshold have to fit together and cannot be optimized separately.

As mentioned before, there are significant differences in color between individual pixels of one color. This can be reduced by smoothing the image. However, this also blurs the color edges so combination colors are created on the border. These are mostly classified as background so that a small stripe between two objects of different color is classified as background when using strong smoothing. Thus the objects are not segmented up to the border so clearly visible to us but a little smaller. If the object border needs to be determined very accurately, one has to classify using weak smoothing, a low rejection threshold and many reference colors. Figure 11.14 shows the effects of weak smoothing. Both images were segmented using the high rejection threshold from Figs. 11.12 and 11.13. In the middle the color selection from Fig. 11.12 was used. One can see that in the middle image only a small fringe of the wires is segmented. Because of the strong smoothing and the small width of the wire, the strong colors from the inner area were weakened and therefore not selected as reference colors. With weak smoothing, these colors must be defined as additional reference colors which

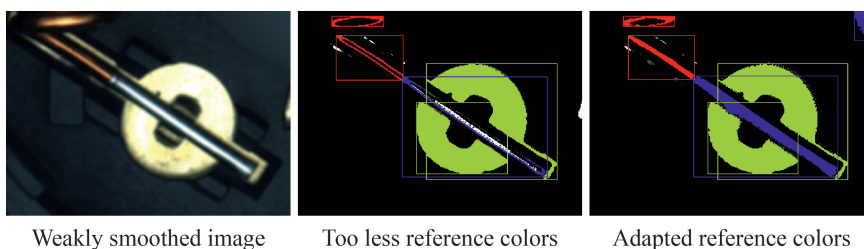


Fig. 11.14 Segmentation with weak smoothing

makes significantly more samples necessary for zinc and copper. The right image of Fig. 11.14 shows the result of a classification with a specially adapted color selection with seventeen zinc, seven copper and eighteen brass samples.

11.2.3 Segmentation

Until now we have merely classified every single pixel in zinc, copper, brass and background but have not yet created any objects. The next step is to combine the connected pixels of one class into objects. This is done just as the creation of object contours in Sect. 4.3.2. The only difference is that in this case not the transition from white to black is searched for but from the current color class to any other color.

Program hint: In **NeuroCheck** color classification of pixels and combining them into objects is done using the check function *Color Matching*. This check function does not use a rejection threshold but a selectivity. A low rejection threshold corresponds to a high selectivity and vice versa. If a large similarity to the reference colors is required, a high selectivity is needed. In so far it behaves like minimum quality in *Template Matching* and *Classify ROIs*.

11.2.4 Presence Verification

The presence verification computation is now easily conducted. After the image capture, the search area is defined. Since there is no prior position adjustment possible, the ROI has to be generously defined. As described above, the image is then smoothed. Because for presence verification a rough estimation of object contours suffices, the image is rather strongly smoothed in this case. For the subsequent segmentation, we only need a smaller reference color set and use a high rejection threshold, as displayed on the right side of Fig. 11.13.

The last step is to check whether the found objects are really the ones we were looking for. The galvanized and the not galvanized wire can be identified rather well using their minimum and maximum Feret's diameter since they are long, narrow objects. Both contact halves can be recognized well because of their area. Because of these features, irrelevant objects such as the first coil of the conductor can be sorted out. Now we have to check that one copper, one zinc and two brass objects are found.

As a final check whether the correct objects were really segmented, the position of the objects with respect to each other can be checked. Because of the smoothing, the segmentation of the two parts cannot reach to where they touch, but the distance between them should be in the range of the smoothing length. Also, the segmented contact halves and the wire do not touch because of the smoothing, but the distance must be in the range of the smoothing length at most. Thus it is

Table 11.4 The following table summarizes the example from Sect. 11.2

Category	Description
Directory	\Examples\Color
Illumination	Front lighting with white light LED
Check functions	The example uses the following check function sequence: 1. <i>Image capture</i> 2. <i>Define ROIs</i> , sets an ROI within which the wire and contact are searched for 3. <i>Smooth ROIs</i> , smoothes the image to reduce color nuances 4. <i>Color matching</i> , classifies the pixels because of their color and combines them to objects 5. <i>Compute features</i> , determines the area and the minimum and maximum Feret’s diameter of the objects 6. <i>Screen ROIs</i> , sorts out all objects except for the contact and the galvanized and not galvanized wire 7. <i>Count ROIs</i> , ensures that all four objects are found 8. <i>Gauge ROIs</i> , determines the geometric arrangement of the objects with respect to each other 9. <i>Check allowances</i> , checks whether the found objects are situated correctly with respect to each other to avoid wrong segmentations

ensured that the correct objects are segmented and the coil wire end is correctly placed on the contact. The entire presence verification is summarized in Table 11.4.

11.3 Summary

Color image processing: There are two typical tasks in color image processing: Color segmentation where objects are found and combined because of their color, and color identification where existing objects are categorized into various color classes.

Color space: Color spaces provide different quantitative descriptions of color. Many color spaces are based on three channels but they differ in the meaning of these channels, the fundamental coordinate system and the intended purpose.

Color classification: In color classification, the ROIs or individual pixels are allocated to a color class. This can be done by comparing the color value with given limits or by a classifier. The first method is suited best when the determination of the color is very stable, e.g. because of averaging an ROI. The second method can be used for classifying individual pixels.

RGB color space: This color space is often used for digital images and also used for monitors and TV. The individual components represent the red, green and blue part of the light.

HSI color space: This color space conforms to the human perception of color. It describes a color through hue, saturation and intensity or brightness. The HSI color space is based on a cylindrical coordinate system.

Template matching: The correlation between a template and an image position can also be computed in a color image. Thus the objects of one color and a specific shape can be searched for. Compared to the search in a gray level image, the computational effort is much higher.

Color segmentation: Color segmentation takes place in two steps: in the first step, all pixels of the ROI are classified according to their color. In the second step, all connected pixels of the same color class are combined into objects.

References

- Gonzalez RC, Woods RE (2008) Digital image processing, 3rd edn. Pearson Education, Upper Saddle River
- Russ JC (2007) The image processing handbook, 5th edn. CRC Press, Boca Raton

Chapter 12

Implementation of Industrial Image Processing Applications

In the preceding chapters we have described a variety of methods and algorithms from the field of image processing and shown their application in various industrial projects. In conclusion, we would like to turn to some aspects that are important for the implementation of industrial image processing applications.

12.1 Image Processing Projects

Firstly, we would like to mention some points that should be observed when implementing a machine vision application in an industrial environment. Some of these correspond to general rules for industrial project management; others are specific to image processing.

Specifications: An image processing system used in production or quality control is nothing other than a special kind of machine. As for any machine, design has to start with a detailed specification of the capabilities of this machine. For the construction of a “purely material” machine, this step is a matter of course. The large part software takes in image processing applications tempts one to neglect careful specifications. For such a flexible and versatile tool as software, however, the risk of a misunderstanding is especially high. Many things can be achieved through software, but this means that one has to be very clear about what is to be achieved—and indeed as early as possible in the design process.

The specifications should always be drawn up by the customer, i.e. the person or company which will ultimately use the system to inspect its products. Of course, the customer will need some help from the image processing specialist, for example advice on what can or cannot be done under what circumstances. But it is very important that the customer has thoroughly considered which properties of a test piece are to be inspected, what tolerances have to be complied with and in what environment the inspection will take place. Even seemingly trivial things like the motion area of a robot arm, the possible assembly dimensions for the lighting setup or the like can have a decisive influence on the final solution. The same is true for production cycle times limiting the choice of the computer system to be

used, communication capabilities for the integration into the production line, etc. Such issues have to be resolved as early as possible.

Sample parts: When the fundamental system requirements have been specified, one can begin to develop a solution. At this stage, a sufficient number of sample parts that reflect the variations inherent in the production process are required. This means that the collection should contain both parts that are unquestionably good and unquestionably faulty as well as borderline cases, i.e. parts which are categorized as borderline-sufficient or borderline-faulty. Recording images of all sample parts to document the solution process is also recommended. These images should be archived in a digital format which can be processed by the image processing system itself. This way the solution can still be shown even if the sample parts are no longer available, or do no longer conform to the specifications of the task because of aging, corrosion etc. It is also advantageous in case of subsequent changes in specifications to be able to demonstrate the agreement with the original specifications.

In some cases it is not possible to obtain an appropriate collection of sample parts, e.g. because the production line is not yet operational. Under such circumstances one has to use all available information, for example preproduction models or laboratory prototypes, to develop a concept leaving sufficient leeway for adaptation. On the part of the image processing system used in the application this requires the capability of reacting to changing conditions on site and carrying out the required modifications without major difficulties.

Illumination and image capture: Creating an optimal lighting setup is always the first step in developing a solution concept. One has to keep in mind that a human can solve a visual inspection task by using his/her entire knowledge and visual facilities that has had millions of years to adjust to the conditions of the real world. A machine vision system, however, has to rely on the relevant information being enhanced in a proper form by illumination. Information not made apparent at this point is lost for good, as well as information destroyed in the process of image capturing and transmission. Setup and quality of illumination and optical components are therefore of critical importance for successful deployment of an image processing system.

Integration into the production line: As with every automated installation, integration of the finished image processing solution into the production requires mechanical, electrical and informational interfaces. Just like the actual inspection task, these must also be agreed upon beforehand. These questions also affect the design of the image processing solution and the selection of illumination and image capturing method. As is usual in engineering work, one has to go through an iterative optimization process.

Problems from the area of mechanics raise questions like the following: Is a separate feeder required? Are the parts to be handled by robots? Can they be stopped during image capturing or do they have to be recorded in motion, etc.

Another very important question is that of information exchange between production process and image processing system. Because of the broad bandwidth

of technologies and their importance for industrial image processing applications, we will take a closer look in Sect. 12.2.

Maintenance: For the operational period of the industrial image processing system, regulations for maintenance and stock keeping of spare parts is required. Image processing systems are automatic installations of high reliability provided that the basis of their work remains stable, i.e. the quality of the captured image remains within acceptable bounds. It has to be ensured therefore that all components function faultlessly within permissible tolerances. If pollution is to be expected due to the production environment, cleaning intervals have to be specified. In harsh environments, all components should be sealed so that they are dust- and waterproof in order to prevent damage or maladjustment of lamps, lenses, cameras, etc., resulting from cleaning and maintenance work on the overall system.

12.2 Process Integration

For industrial image processing it is very important to exchange data with manufacturing control. Therefore, the control system must be able to control the image processing station as part of the manufacturing process and, vice versa, the results gained from image processing must be transferred to the control system. For the data transfer various technologies are used. We will briefly present the most important technologies:

Digital I/O: Transfer of the data takes place over individual signal cables that represent single bits by their voltage level. The advantage of this technology is the minimal latency period. The disadvantage is the limited bandwidth since frequently only 16 or 32 bit can be transferred. Since the cables need an individual lead for each bit, they are rather expensive and sensitive. A distinctive feature of digital I/Os in addition to information transfer is their ability to switch individual illumination elements from the image processing computer. To access a digital I/O interface from the PC, special expansion cards are used.

Fieldbus: From a user's perspective a fieldbus seems to be state-based just like a digital I/O. Actually, the information is exchanged serially via a network. These messages are exchanged on a clock cycle so usually the information is updated with a delay. The advantage of fieldbus compared to digital I/O is that with every data exchange packages of several 100–1,000 bytes can be exchanged. Furthermore, the wiring is much more cost effective.

For use in industrial manufacturing, a number of different protocols have established themselves such as PROFIBUS, CC-Link, CANopen or DeviceNet. To integrate a PC into a fieldbus network, an appropriate expansion card must be installed in the PC.

Industrial Ethernet: While fieldbuses of the first generation exchanged data via serial interfaces, fieldbuses of the second generation use Ethernet. That is why

this technology is often called Industrial Ethernet; however, the use of this term is not unambiguous.

Compared with serial data transfer, Ethernet allows for a significantly larger amount of data to be transmitted. However, to use Ethernet as a fieldbus medium has the disadvantage that the transmission times are not overly deterministic. The various Industrial Ethernet protocols such as PROFINET, EtherNet/IP, Ethernet Powerlink or EtherCAT frequently extend the Ethernet standard to achieve sufficient real-time performance. Some of these extensions have to be implemented in hardware requiring the installation of an expansion card in the PC despite an integrated standard Ethernet interface.

While the various fieldbus protocols have been established for years, the market for Industrial Ethernet is still in flux. However, it's realistic to expect a limited number of protocols to become accepted by the market.

Serial interface: Serial interface usually denotes an RS-232 interface that used to be utilized to handle communication between the PC and its peripherals. Over the last years they were replaced primarily by USB. So, nowadays there is hardly a PC with a default serial interface.

The advantages of serial interfaces used to be low cost and the large amount of data (at least for those times). In industrial communication serial communication is being gradually replaced by Ethernet that is supplied with almost every PC.

There is no uniform format for the exchanged data; instead this is individually agreed upon between the control programmer and the developer of the image processing application.

Network: The de facto standard for network technologies is Ethernet. Since almost every new PC is equipped with an Ethernet interface, data exchange via Ethernet is very common today. Unlike Industrial Ethernet, other protocols from Internet technology are used such as the well-known IP and TCP. Ethernet is characterized mainly by the large amount of data that can be transferred. However, long delays during data transmission can occur with Ethernet, especially if several users are sending data over one network simultaneously.

Files: Files are not a transmission technology. Actually, the exchange of files between the image processing computer and the manufacturing control takes place via an Ethernet network. The classification as transmission technology follows (Hornberg 2006). After all, files are still an important medium to concentrate and exchange large amounts of data. Since the exchange of files between two network parties takes place via a standard functionality of the operating system, this is relatively easy to implement.

The data formats of files are as diverse as their content. While for the storage of images standard formats like TIFF, BMP, PNG etc. are frequently used, proprietary formats are usually used for files with gauging values and reading results. For saving alphanumerical data, there is a clear trend to XML formats since they can be read both by computers and humans. Files also comprise data bases where results or inspection-specific parameters can be stored.

Information classes: Apart from the transmission medium, the information exchange can also be classified using the information type. (Hornberg 2006)

distinguishes three information classes according to range and intended use of the information:

- Control signals
- Results and parameters
- Mass data.

Besides the content, the various classes differ in range and the technology used for transmission.

Control signals: The industrial image processing system has to be synchronized with the manufacturing process; for this control signals are exchanged between the process control and the image processing computer. Commands are sent from the control system to the image processing computer. From the point of view of the image processing computer these are input signals. Typical input signals are:

- *Start inspection*, tells the image processing system to execute the next inspection.
- *Change inspection type*, changes the inspection if, for example, a part of a different type needs to be inspected.
- *Abort inspection*, aborts the current inspection run.
- *Change user*, changes between users with different rights so certain configuration changes can only be done by properly trained personnel.

The image processing computer sends its status and the status of the inspection to the control of the manufacturing process. For the image processing computer, these are output signals. Typical examples are:

- *System ready*, the image processing system is waiting for the next command.
- *System busy*, the image processing system is executing a command; i.e. usually an inspection is running.
- *Inspection OK*, the last part inspected was OK.
- *Inspection not OK*, the last part inspected was not OK.

Moreover, the messages *Inspection OK* and *Inspection not OK* are frequently used to determine when the result of the image processing system is available.

Since the control signals are used to synchronize the image processing system and manufacturing process temporally, they have to occur with a minimal latency period. For this reason, field bus, digital I/O or Industrial Ethernet are frequently used as transmission technology. With slower systems, networks or the serial interface may be used as well. The amount of data transmitted with control signals is usually low so many can be transmitted using individual bits, which makes the use of digital I/Os possible in the first place.

Results and parameters: In many cases it doesn't suffice that the image processing systems sends the final result of the inspection to the control computer. Often, additional results such as measuring values, reading results, object positions etc. have to be transmitted to the control system. This data may be used for very different purposes such as long-time archiving or statistical evaluation.

With many products there are minor changes between batches such as a change of color, type number, some dimensions or the position of individual inspection features of the inspection object. When these changes are minor, it is not worthwhile to develop an individual inspection for each type; rather, it is more cost-effective to change the individual parameters of the inspection. Thus relevant parameters have to be transmitted from the control system to the image processing computer.

The amount of data of results and parameters starts with a couple of bytes and can grow to any size for complex parameterization. Depending on the amount of data, different transmission technologies are used. For small amounts of data a fieldbus or Industrial Ethernet will suffice; larger amounts of data can be transmitted via Ethernet, while the exchange of very large amounts of data takes place mainly via files. The bandwidth of a digital I/O is usually too small to transmit results or parameters.

Mass data: With image processing systems, mass data means images. In some image processing stations, all images captured must be stored for long-term archiving. This way, if there are any complaints, it can be proved that the inspection part had left the manufacturing level—at which the inspection had taken place—flawlessly. Other stations, for example, send the images of faulty parts to a station for rework. Using the image, the worker can see which feature is flawed and may be able to rectify it. In such cases, additional information can be artificially overlaid in the images.

Since for images the amount of data is frequently in the range of one or several megabytes, only files or networks can be used as the transmission media.

12.3 Outlook

Over the past years, a trend towards ever more challenging industrial image processing tasks can be observed. The challenge itself is not only the image evaluation proper. Often, a comprehensive, graphic visualization of the inspection results and process data is requested. Furthermore, various data must be exchanged with the superior control system.

To meet all these requirements, image processing systems need a flexible and powerful architecture. PC systems with modern multithreading-capable operating systems such as Microsoft® Windows® 7 can be adjusted quickly to various requirements because of their flexibility and expandability. In addition, new technologies are often available for PCs at an early stage. For example, image processing in particular benefits extremely from modern multi-core processors since many of the algorithms used can be distributed over several processor cores with suitable implementation. The change to 64-bit technology has resulted in another performance boost especially for memory-intensive line-scan camera applications.

Using PC hardware and operating systems also facilitates service and maintenance by using proven standard components. Standardization also makes it easier for on-site personnel to operate and configure the systems because they can use experience from daily work with PCs in other areas. Especially with frequently changing requirements, the ability of the operating personnel to adjust the image processing system through configuration can be vital for the performance.

Reference

Hornberg A (2006) Handbook of machine vision. Wiley-VCH-Verlag, Weinheim

Appendix A

Mathematical Notes

A.1 Backpropagation Training

In this section we will describe the backpropagation training method for neural networks of the multilayer perceptron type described in Sect. 6.4.2. But before we start, we would like to say some things about the history of this network type.

A.1.1 Neural Networks: Concept and History

Neural network, connectionist system, parallel-distributed processing—all these terms denote systems trying to solve certain information processing tasks by connecting a multitude of processing units which are—viewed in isolation—quite simple. The computational power of the system is derived from the complexity of the connectivity structure of these processing units.

The most powerful known neural network is the human brain. From a functional—not biological—point of view, a biological neuron is a rather simple thing compared, for example, to a modern computer processor. Nevertheless, the human brain is capable of feats of information processing impossible for computers, especially in the area of pattern recognition.

Compared to the brain, currently existing technical systems based on this paradigm are rather modest, in size as well as in capabilities. But still, they have remarkable potential in certain areas. To differentiate from biological neural networks—like the brain—these technical systems are usually referred to as *Artificial Neural Nets* (ANN).

A first milestone in the development of artificial neural networks was the formal description of a neuron by McCulloch and Pitts in McCulloch and Pitts (1943). This description of a neuron as a simple thresholding element was rather primitive, but it did already contain a very important property, that of nonlinearity, and was capable of illustrating the potential of neuron clusters for information processing.

In comparison to biological neural networks, a very important element was still missing: the capability to learn from experience. In Hebb (1949), a method was introduced to incorporate the capability of adaptation into artificial neural networks. Variations of this original learning rule are still used today.

Rosenblatt (1958) introduced the perceptron as the first self-contained model. Hereafter, interest in neural networks waned since it turned out that the original goals were not as easily achieved as researchers had hoped. It wasn't until the 1980s that significant progress was made, including the work of Hopfield (1982), on auto-associative networks, Teuvo Kohonen on self-organizing networks (Kohonen 1982), and—most important for our technical purposes—the development of the generalized delta rule by Rumelhart and Hinton (Rumelhart and McClelland 1986).

Besides these new theoretical developments, progress in information technology was of prime importance for the renewed interest in artificial neural networks. Only the increased computation performance and memory capacity of modern computers allowed a serious evaluation of neural networks for real-world application problems.

Neural networks have some desirable properties for industrial applications, namely:

- *Robustness*: Neural networks do not react with sudden failures to distorted or noisy input signals. Instead, their performance decreases gradually (so-called *graceful degradation*). This makes them well suited for pattern recognition tasks where variations or distortions of the patterns are to be expected. Neural network hardware solutions can also be designed to be robust with respect to failure of individual units since the “knowledge” of the network is not located in individual units but distributed over the entire connectivity structure.
- *Parallelization*: In principle, all processing units of a network can work simultaneously on a problem.
- *Adaptivity*: Denotes the ability of neural networks to change their internal state using training examples in order to acquire the capability of solving certain tasks without requiring an explicit formulation of the solution algorithm. This process is often called “learning”, but its capabilities are still far inferior to learning in biological brains.

A.1.2 Basics

This section briefly reviews the basic workings of the multi-layer perceptron network from Sect. 6.4.2 so that we have all the information available required to explain the training method. Figure 6.5 shows the typical structure of a multi-layer perceptron. It consists typically of three layers of processing units, an input layer, a so-called hidden layer and an output layer. Between those layers of units there are two layers of interconnections.

The activation state or simply activation of a unit j is a function of the weighted sum of the outputs o_k of all units k connected by weighted connections w_{kj} with unit j :

$$a_j = f_a \left(\sum_k w_{kj} o_k - \theta_j \right). \quad (\text{A.1})$$

The most common activation function is the *logistic function* given by

$$f_a(\text{net}_j) = \frac{1}{1 + e^{-(\text{net}_j - \theta)}}. \quad (\text{A.2})$$

Advantages of the logistic function are that it suppresses output noise through its asymptotic behavior and its simple derivative:

$$f(x) = \frac{1}{1 + e^{-x}}, \quad (\text{A.3})$$

$$\frac{df}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2}, \quad (\text{A.4})$$

$$\frac{df}{dx} = f(x) (1 - f(x)). \quad (\text{A.5})$$

Practical implementation will almost always use a look-up table for the values of the logistic function to avoid the time required for the computation of the exponential function. Because of the simple form of the derivative its values can be taken from the same look-up table so that the derivative required during training can be computed very rapidly.

A.1.3 Backpropagation

Backpropagation is a general method for continuous optimization problems. It works using gradient descent and has gained enormous popularity as training method for multi-layered networks of the perceptron type shown in Fig. 6.7. Strictly speaking, it should be called *generalized delta rule*, but the term backpropagation is commonly used. The backpropagation algorithm will not be formally derived here but we will only present the basic idea. The derivation can be found in most books on neural networks, for example in Zell (1994).

Training of the original single-layer perception is conducted in a supervised fashion. An input pattern with a known target output is presented to the network. The state of the output layer generated by this input is compared to the target output. If it deviates from the target output, the connection weights of the network are changed correspondingly in order to minimize the deviation. Unfortunately,

this rule cannot be applied directly to multi-layered networks because no target outputs are known for their hidden layer(s). The solution to this problem was found with the *generalized delta rule* with *error backpropagation* (Rumelhart and McClelland 1986).

The goal of the algorithm is to minimize the sum of the errors occurring during classification of all training patterns. For the classification of a single pattern, the error is defined as

$$F_m = \frac{1}{2} \sum_j (t_j - o_j)^2. \quad (\text{A.6})$$

with m as the number of the presented training pattern; t_j as the desired output value of output unit j , o_j as the actual output value of the same unit.¹ The goal is to reach the global minimum of the sum of errors for all training patterns,

$$F = \sum_m F_m.$$

To reach this minimum, the weights are always changed in the direction of decreasing error, i.e. in opposite direction to the gradient of the error with respect to the weight:

$$\Delta w_{ij} \propto -\frac{\partial F}{\partial w_{ij}} \quad (\text{A.7})$$

This is the original *delta rule*. It works fine for adapting the input weights of the output units for which desired output values are known. But it does not tell us anything about the input weights of hidden layers for which no desired output—and hence no error term—is known. Backpropagation as an extension of the delta rule is based on the fact that the output of the final layer—and therefore the error F_m —is a function of the input values of the final layer, which in turn are functions of the outputs of the preceding—hidden—layer of the network and the connection weights between these two layers. The output values of the hidden layer, and this is the essential idea, are a function of the weights connecting input and hidden layers, leading to the conclusion that the error F_m also is a function of these weights. It is therefore possible to define an error term for the hidden layer(s) using partial differentiation according to the chain rule. From this error term, the gradient for adapting the input weights can be determined.

The equation for adapting the weight w_{ij} connecting a hidden unit i and an output unit j of the network is with η as the learning rate, f'_a as the derivative of the activation function, o_i as the output value of the hidden unit:

$$\Delta w_{ij} = \eta \delta_j f'_a(\text{net}_j) o_i. \quad (\text{A.8})$$

¹ Strictly speaking, these values should also have an index m for the number of the current input pattern. For clarity's sake, this index has always been left out in the following, since all equations deal solely with the error for a single input pattern.

This error term is given by δ_j :

$$\delta_j = -\frac{\partial F}{\partial net_j} = (t_j - o_j)f'_a(net_j). \quad (\text{A.9})$$

For the hidden units, a corresponding error term δ_j is derived from the error term of the output units as follows:

$$\delta_j = f'_a(net_i) \sum_j \delta_j w_{ij}. \quad (\text{A.10})$$

With this new error term, the weight update for the hidden units again follows Eq. (A.8).

In essence, this means that first the error values for the output units are computed; from these the error terms for the hidden units are then computed. This is also the reason for the name *error-backpropagation* since the error propagates backwards from the output units through the network.

Training procedure for backpropagation: The form of the backpropagation algorithm described above tries to minimize the sum error over the entire set of training patterns, i.e. it processes all patterns in the set before computing the error values and updating the weights. This is called *batch backpropagation*.

In practice, learning is usually performed sequentially. This means that instead of using the derivative of the sum error, the weights are updated after processing of each training pattern. This method is called *standard* or *online backpropagation*. The weight update then follows from

$$\Delta w_{ij} \propto -\frac{\partial F_m}{\partial w_{ij}} \quad (\text{A.11})$$

using the partial derivative of the error for a single pattern as a momentary estimate of the sum error.

For every pattern from the training set the following sequence is carried out:

1. *Forward pass*: an input pattern is presented to the network; the activation values are propagated from layer to layer until the output pattern is created as the vector of output values o_j of the units of the final layer.
2. *Error backpropagation*:
 - The output pattern is compared to the target output; for each output unit the error term is computed according to Eq. (A.9). From this error term, the weight updates for the input weights to the output units follow from Eq. (A.8).
 - For the penultimate layer, i.e. the hidden layer immediately preceding the output layer, error terms are computed from those of the output layer, using Eq. (A.10). From these error terms, the weight updates for the input weights of this layer follow in the same way as for the output layer.
 - Error terms and weight updates for further hidden layers are also computed from Eqs. (A.10) and (A.8), but instead of the error terms of the output units in Eq. (A.10) the error terms of the subsequent hidden layer have to be used.

This procedure is repeated until the minimum of the error is found. In practice one will define a limit value for the sum error or a maximum number of passes through the training set.

Problems and extensions of backpropagation: Backpropagation has proven in many applications to be a very successful training method. It is not free of drawbacks, however, for which various remedies have been suggested. A very important problem is the influence of the learning rate. If a rate which is too large is chosen, the network may jump over the error minimum in an optimization step and start to oscillate. If it is chosen sufficiently small to avoid this problem, training time may become impractical. The *momentum* term suggested in Rumelhart und McClelland (1986) can alleviate this problem. The equation for the weight update is changed to

$$\Delta w_{ij}(t+1) = \eta o_i \delta_j + \alpha \Delta w_{ij}(t). \quad (\text{A.12})$$

The effect is that the weight change is not only based on the current error gradient but also—up to a certain degree given by the momentum term—on the error gradient from the previous step. This means that the network prefers to continue in its previous direction. The effect of this changed weight update equation is to accelerate training on large plateaus of the error surface and to slow it down in fissured areas. Typically, the momentum term α is set to values between 0.2 and 0.99.

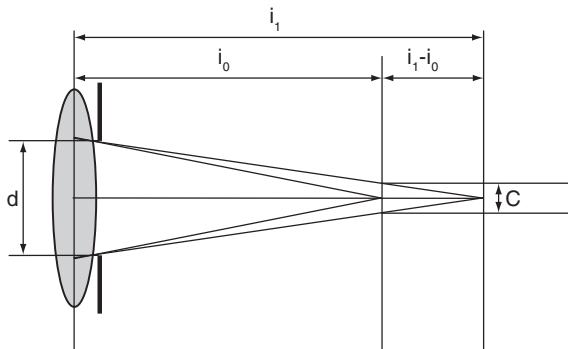
Another problem is created by units working in the nearly horizontal area of the sigmoidal activation function because there the derivative of the activation function is practically zero, slowing down backpropagation as a gradient-based method almost to a standstill. This is called saturation. In Fahlman (1988), it is suggested simply to add a constant value of 0.1 to the derivative of the activation function. This is denoted as flat spot elimination, since it removes the influence of the flat area of the activation function.

A.2 Computation of the Depth of Field

In this section, the formulas for the computation of the depth of field used in Sect. 8.7.3 are derived.

A.2.1 Limit Distances

Figure A.1 shows the creation of a circle of confusion in the image plane. We assume that the lens is set to the object distance o_0 where the image is created at a distance i_0 from the lens. Now the object is moved closer to the lens, up to an object distance of $o_1 < o_0$. According to the thin lens imaging Eq. (8.8), image distance increases with reduced object distance. A focused image of the object is

Fig. A.1 Creation of a circle of confusion

only achieved for a distance $i_1 > i_0$ from the lens. If the setting of the lens is not changed, the image plane is still at a distance i_0 from the lens so there is no image at that point but a circle of confusion.

The size of the circle of confusion follows from the intercept theorem

$$\frac{C}{d} = \frac{i_1 - i_0}{i_1}, \quad (\text{A.13})$$

$$C = \frac{i_1 - i_0}{i_1} d. \quad (\text{A.14})$$

Expressing the aperture through the f-number k (see Sect. 8.5.1) and the focal length f by

$$d = \frac{f}{k}. \quad (\text{A.15})$$

we get

$$C = \frac{i_1 - i_0}{i_1} \frac{f}{k}. \quad (\text{A.16})$$

According to the thin lens imaging Eq. (8.8), the image distance i_i of an object can be expressed by the focal length f and the object distance o_i in the following way:

$$\frac{1}{i_i} = \frac{1}{f} - \frac{1}{o_i} = \frac{o_i - f}{fo_i}, \quad (\text{A.17})$$

$$i_i = \frac{fo_i}{o_i - f}, \quad (\text{A.18})$$

Inserting this for i_0 and i_1 in Eq. (A.16), it follows:

$$C = \frac{\frac{fo_1}{o_1 - f} - \frac{fo_0}{o_0 - f}}{\frac{fo_1}{o_1 - f}} \frac{f}{k}, \quad (\text{A.19})$$

$$C = \frac{fo_1(o_0 - f) - fo_0(o_1 - f)f}{fo_1(o_0 - f)} \frac{f}{k}, \quad (\text{A.20})$$

$$C = \frac{fo_0 - fo_1}{o_1(o_0 - f)} \frac{f}{k}, \quad (\text{A.21})$$

$$C = \frac{f^2 o_0 - f^2 o_1}{ko_1(o_0 - f)}. \quad (\text{A.22})$$

Thus we get the diameter of the circle of confusion as a function of the set object distance o_0 and the actual object distance $o_1 < o_0$. It is obvious that the diameter grows the smaller o_1 is. This can be seen in that the image is more and more out of focus the closer the object is placed in front of the lens. This poses the question how close the object can be brought to the lens without a *perceivable* blur. The criterion for this is the diameter of the circle of confusion. If this is sufficiently small, the blur is viewed as not perceivable. For each diameter C of the circle of confusion, the object distance o_1 can be calculated from Eq. (A.22) and thus the minimum permissible object distance o_{min} for a maximum permissible diameter C^* :

$$o_1 = \frac{f^2 o_0 - f^2 o_1}{Ck(o_0 - f)}, \quad (\text{A.23})$$

$$o_1 = \frac{f^2 o_0}{f^2 + Ck(o_0 - f)}, \quad (\text{A.24})$$

$$o_{min} = \frac{f^2 o_0}{f^2 + C^*k(o_0 - f)}. \quad (\text{A.25})$$

Now the smallest distance to the lens is known where an object is still imaged in focus. In the opposite case—i. e. when the object is farther away from the lens than the nominal distance—one can proceed as before, taking into account that the new distance o_1 is larger than the nominal distance. This means that now the image will be closer to the lens, i.e. $i_1 < i_0$. Since we are only interested in the absolute values of the distances, we simply reverse the difference in Eq. (A.14). Trigonometrically, nothing changes due to the similarity of the triangles. Thus we find for the diameter of the circle of confusion:

$$C = \frac{i_0 - i_1}{i_1} \frac{f}{k}, \quad (\text{A.26})$$

and after a similar transformation the upper limit distance becomes

$$o_{max} = \frac{f^2 o_0}{f^2 - C^*k(o_0 - f)}. \quad (\text{A.27})$$

From Eqs. (A.25) and (A.27) for the lower and upper limit distances the depth of field follows as

$$\Delta o = o_{\max} - o_{\min} = \frac{f^2 o_0}{f^2 - C^* k(o_0 - f)} - \frac{f^2 o_0}{f^2 + C^* k(o_0 - f)} \quad (\text{A.28})$$

$$= \frac{2f^2 o_0 C^* k(o_0 - f)}{f^4 - C^{*2} k^2 (o_0 - f)^2}. \quad (\text{A.29})$$

The acceptable unsharpness C^* is a rather small value. If one uses half the dimensions of a sensor element, C^* is in the area of five micrometers, thus C^{*2} is approximately $2.5 \times 10^{-5} \text{ mm}^2$. As long as o_0 is not considerably larger than the focal length f , the second term in the denominator does not carry as much weight as f^4 so for close-ups the Eq. (A.29) can be simplified according to Ernst (1991) like this:

$$\Delta g = \frac{2C^* k o_0 (o_0 - f)}{f^2}. \quad (\text{A.30})$$

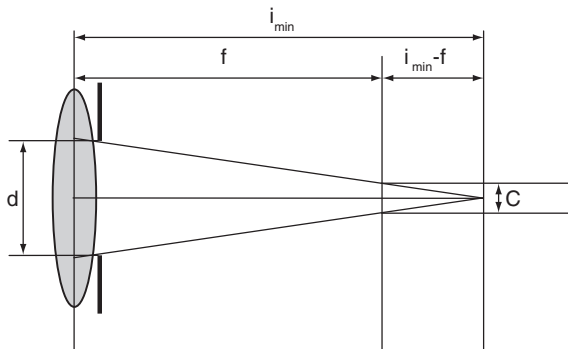
A.2.2 Depth of Field at Infinite Distance

The approximation in Eq. (A.30) can result in considerable errors for large object distances. On the other hand, the denominator in Eq. (A.29) will become negative for large values of o_0 which means that the depth of focus would have a range of negative size. Correspondingly, the upper limit distance in Eq. (A.27) will become negative for the limiting case of infinite nominal distance $o_0 \rightarrow \infty$. The reason is that the equations have been derived for a circle of confusion having the maximum acceptable diameter. In the limit case of infinite distance, all light rays from the object will be parallel and therefore will always be focused precisely in the focal plane (this, of course, is the reason why from a certain distance onward all objects appear sharp in the image). This means we have to pose the question a little differently, asking from what distance o of an object the circle of confusion within the focal plane of the lens will be smaller than the acceptable diameter.

Figure A.2 again shows the creation of a circle of confusion under the assumption that the lens has been set to form the image precisely in its own focal plane. The nominal image distance i_0 is then equal to the focal length f . An object which is too close to the lens will be in focus behind the focal plane, i.e. it will create a circle of confusion of diameter C in the focal plane. The diameter of this circle of confusion follows from Eq. (A.16) by inserting the focal length for the nominal image distance i_0 :

$$C = \frac{i_{\min} - f}{i_{\min}} \frac{f}{k}. \quad (\text{A.31})$$

Fig. A.2 Circle of confusion when focusing on the focal plane



Expressing the image distance through the object distance according to the law of imaging by

$$i_{min} = \frac{f o_{min}}{o_{min} - f}, \quad (\text{A.32})$$

we find for the diameter of the circle of confusion in the focal plane depending on the object distance o_{min} :

$$C = \frac{\frac{f o_{min}}{o_{min} - f} - f}{\frac{f o_{min}}{o_{min} - f}} \frac{f}{k}, \quad (\text{A.33})$$

$$C = \frac{f^2}{o_{min} k}. \quad (\text{A.34})$$

The necessary distance for a sharp image follows from the requirement that this diameter has to be smaller than C^* as:

$$C^* > \frac{f^2}{o_{min} k}, \quad (\text{A.35})$$

$$o_{min} > \frac{f^2}{C^* k}. \quad (\text{A.36})$$

This means that each object which is at a distance of at least $f^2/(C^* k)$ from the lens will be imaged with an unsharpness of less than C^* in the focal plane. Again using half the size of a sensor element as the acceptable unsharpness, i.e. approx. five micrometer, results for a 12 mm lens and an f-number of 11 in a limit distance² of 2,618 mm.

² The value for the upper limit distance in Eq. (A.27) becomes negative at a larger distance because C is fixed in this equation but will in reality shrink with increasing distance. The accuracy of the formula is sufficient for all practical purposes.

A.2.3 *Dependence of the Depth of Field on the Focal Length*

It can be easily shown that increasing the focal length will increase the lower limit distance. If the focal length is increased by the value Δ , it follows from

$$o_{min}(f + \Delta) > o_{min}(f), \quad (\text{A.37})$$

$$\frac{(f + \Delta)^2 o_0}{(f + \Delta)^2 + C^*k(o_0 - (f + \Delta))} > \frac{f^2 o_0}{f^2 + C^*k(o_0 - f)}, \quad (\text{A.38})$$

after some transformations

$$2f o_0 - 2f^2 + \Delta o_0 > 0. \quad (\text{A.39})$$

We had required that the focal length is increased, i.e. $\Delta > 0$. Since we are computing using absolute values, it also holds that $o_0 > 0$. In any case, the last summand is positive. Since within the focal length a focused image is not possible, the set object distance o_0 is larger than the focal length f , thus the difference $f o_0 - f^2$ is also positive. Inequation Eq. (A.39) is therefore always true, hence the original inequation Eq. (A.38) must also be true. This shows that when increasing the focal length, the minimum object distance for a focused image also increases. Similarly, it can be shown that the upper limit distance will decrease with increasing focal length. This explains the small depth of field range of zoom lenses with their large focal lengths.

Appendix B

Software Download

All examples in this book and the image processing software **NeuroCheck**, which is used for the examples and industrial applications described, are available for download at <http://extras.springer.com/> of Springer Publishing. The software works with the operating systems Windows[®] 8, Windows[®] 7, Windows[®] Vista[®] and Windows[®] XP by Microsoft. To use the software, you will need .NET framework version 3.5. This is not part of the download but may be downloaded free of charge from the Microsoft website (www.microsoft.com). Because of the dynamic development in the software area, we are planning on updating the software and other data in the download on a regular basis. Therefore, the requirements listed above may change. For further information about the content and structure of the download, please refer to the ReadMe.txt file.

In the demo version of NeuroCheck supplied with the download, the entire image processing functionality can be used without limitations. Thus all the examples described in the book can be reproduced and further experiments conducted using the supplied images. The functional limitations of the demo version mainly concern using the software for designing and operating automated visual inspection systems.

All examples described in the book can be found in the “Examples” directory of the download. The tables describing the main application examples throughout the book list the subdirectories where you will find the data for each example.

For further information on the NeuroCheck software and its applications, please turn to the website of NeuroCheck GmbH <http://www.neurocheck.com>.

Appendix C

Weblinks to Industrial Image Processing

The weblinks listed here are just a small selection of websites dealing with the subject of machine vision. It was our aim to present the variety of companies and organizations dealing in some way with industrial image processing, not to give a representative or even remotely comprehensive list.

- <http://ibv.vdma.org>
Department of Industrial Image Processing, German Engineering Association (Verband Deutscher Maschinen- und Anlagenbau e.V. (VDMA, Frankfurt am Main, Germany))
- www.emva.org
European Machine Vision Association (EMVA, Barcelona, Spain)
- www.visiononline.org
Automated Imaging Association (AIA, Ann Arbor, MI, U.S.A.)
- www.neurocheck.com
NeuroCheck GmbH (Remseck, Stuttgart, Germany)
- www.industrialvision.co.uk
Industrial Vision Systems Ltd. (IVS, Kingston Bagpuize, UK)
- www.baumer.com/vision/
Baumer Optronic (Radeberg, Germany)
- www.baslerweb.com
Basler AG (Ahrensburg, Germany)
- www.dolan-jenner.com
Dolan-Jenner Industries (Boxborough, MA, U.S.A.)
- www.messe-stuttgart.de/vision/
Trade show VISION, Landesmesse Stuttgart GmbH (Stuttgart, Germany)

Index

0–9

64-bit, 344

A

Aberration, chromatic, 248
Aberrations, 242, 248
Achromat, 248
Activation function, 169
Activation state, 165
Active pixel sensors, 208
Adaption rule, 166
ADC. *See* Analog-to-digital converter, 207
AEC, 218
AGC, 218
Analog-to-digital converter, 207, 209
Angle increment, 82
Angle measurement, 183, 184, 187, 200
Angular increment, 309
Aperture, 198, 238, 243, 245, 247, 248, 262
Apochromat, 248
Area, light-sensitive, 205, 209, 210, 213, 214
Area. *See* Object features, area, 304
Artifact, 89, 281
Axial/in-line illumination, 257
Axis computation, 71, 72

B

Backlight, 174, 183, 254, 258, 260
 directed, 258, 264
 polarized, 272
 telecentric, 195, 258, 264
Back lighting, 16, 68
Back lighting. *See* Transmitted light, 187
Backpropagation, 166, 171
Bar code identification, 119, 149
Bayer pattern, 219
Bayes classifier, 157, 171
Bayes' theorem, 156

Best-fit circles, 193
Binning, 219
Blooming, 216
Borderline samples, 126, 167
Bright field illumination. *See* Illumination, bright field, 255
Brightness distribution, 315
Brightness variations, 253

C

Calibration factor, 176
Calibration mode, 197
Calibration target, 198, 200
Caliper measurement, 181, 182
Camera, color, 219
Camera
 digital, 200, 217, 261
 intelligent, 8
 line-scan, 231, 262
 output format, 221
 pan/tilt/zoom, 266
 single-chip color, 212, 219
 three-chip color, 212
 three-layer color, 213
CameraLink, 223, 224, 232, 261
CCD sensor, 204, 206, 216, 261
Center fiber, 313, 317
Centroid coordinates, 199
Centroid distance, 179
Centroid. *See* Object features, Centroid, 308
Chain code, 95, 124, 128, 305, 306, 317
Character recognition, 125, 153, 170
Charge amplifier, 207, 208, 216
Chromatic aberration. *See* Aberration, chromatic, 248
Circle computation, 194
Circle of confusion, 246
Class border, 162
Class boundary, 171

Class index, 154, 155
 Class space, 152, 153, 158, 170
 Classification, 172
 Classification application, 130
 Classification function, 153, 158, 170, 172
 Classification process, 167, 168, 171, 172
 Classifier application, 124, 137, 149, 151, 170, 319
 Close up, 179, 250
 Closing, 57, 62
 CMOS sensor, 204, 208, 216, 261
 C-mount. *See* Flange back length, 249
 Coarse-to-fine-search, 99
 Cold light source, 254, 257, 263
 Collimator lens, 258
 Color balance. *See* White balance, 220
 Color channel, 211, 219
 Color description, 324
 Color filter array, 219
 Color filter mosaic. *See* color filter array, 212
 Color identification, 320
 Color image processing, 319
 Color segmentation, 332, 338
 Computation of center of gravity, 68, 69
 Computation primary axis, 73
 Contour distances, 182, 200
 Contour representation, 95, 110, 304, 314
 Contour tracing, 66, 92–94, 110, 124, 129, 138, 150, 287
 Contrast enhancement, 29, 30, 31, 34, 61, 129, 150
 local, 33
 Contrast filter. *See* Filter, contrast, 57
 Contrast of the image, 29
 Contrast range. *See* Dynamic range, 205
 Control signals, 343
 Contour distance, 181, 200
 Convex object, 179
 Coordinate transformation, 58, 62
 Correlation coefficient, 96, 97, 99, 104, 110, 171, 285, 301
 Correlation points, 101
 Cost function, 157
 CS-mount. *See* Flange back length, 249
 Curvature. *See* Object features, Curvature, 310

D

Dark current. *See* Noise, thermal, 222
 Dark field illumination. *See* Illumination, dark field, 256
 DCAM, 227

Decision check function, 156, 157, 162, 163
 Decision function, 169, 171
 Demosaicing, 219, 261
 Depth of field, 262
 Difference image, 146, 147, 150
 Digital I/O, 341
 Digitization, 10
 Dilation, 56, 62, 140, 281, 286, 287
 Dimensions, 306
 Dimensions. *See* Object features, dimensions, 303
 Discretization, 92, 118, 306, 307, 309, 310
 Discriminating hyperplanes, 168
 Distance measuring, 74, 181
 Distance variations, 195
 Distance-versus-angle signature, 75, 78, 82, 310, 312
 Distant objects, 247
 DMA, 225
 Dynamic range, 205, 222

E

Edge detection, 111, 132, 188, 190, 200
 Edge direction, 107
 Edge image, 37, 106, 111
 Edge model, 107, 109, 111, 200
 Edge probing, 106, 108, 111
 Edge segmentation, 109
 Edge slope, 50
 Enclosing rectangle. *See* Object features, Enclosing rectangle, 303
 Endoscope, 252, 263
 Erosion, 55, 62
 Error rate, 129, 158, 171
 Ethernet, 342
 Euler number. *See* Object features, Euler number, 313
 Exposure. *See* Exposure time, 218
 Exposure time, 143, 206, 208, 209, 218, 237, 238
 Extension ring, 249, 250, 263

F

False edges, 42
 Feature scaling, 168, 172
 Feature space, 129, 152, 157, 159, 162, 170
 Feature vector, 154, 156, 157, 161, 162, 171
 Feret's diameter. *See* Object features, Feret's diameter, 309

Fiber length. *See* Object features, fiber length, 312

Fiber optic light guide, 254, 257, 263

Fiber width. *See* Object features, fiber width, 313

Field, 207

Fieldbus, 341

Files, 342

Fill factor, 214

Filter, 70, 111, 150

- adapted, 46
- border effect, 42
- contrast, 57, 62, 139, 299, 328
- difference, 57, 139, 140
- edge, 52, 62
- Gaussian, 48
- Laplacian, 51
- linear, 52
- low pass, 48
- mean value, 40, 46, 52, 53, 62, 299
- median, 53, 62
- morphological, 57, 62
- rank, 53
- rectangular, 46, 52, 62
- smoothing, 62
- Sobel magnitude, 52
- sparse, 46

Filter kernel, 57, 62

Filter kernel. *See* Filter mask, 39

Filter mask. *See* Filter mask, 39

Filter operator, 53, 54

FireWire, 224, 226, 262

Fixed pattern noise, 209, 222

Flange back length, 249, 263

Fluorescent tubes, 254, 257, 263, 281

F-number, 238, 245, 262

Focal length, 242, 245, 247, 248, 250, 262

Focal plane, 240, 244, 251

Focus, 238, 251

Form factor, 307

Format, optical, 214

Fourier descriptors. *See* Object features, Fourier descriptors, 314

Fourier transform, 60

Frame grabber, 223, 224

Frame rate, 218

Front light

- diffuse, 255

Front lighting, 16, 138, 188, 255, 263

- directed, 255, 263, 323

Full well capacity, 205

Functional-link classifier, 171

Function-based classifier, 163

G

Gain. *See* Signal amplification, 218

Gamut, 213

Gauging, 10, 75, 106, 251, 256–258, 264, 289, 301, 201

Gigabit Ethernet, 224, 229, 262

GigE Vision, 229

Global shutter, 210

Gray level gradient, 90, 190

Gray level quantization, 11, 306

Gray level statistics. *See* Object features, gray level statistics, 11, 315

Gray scale transformation, 25, 33, 61

H

Halogen lamp, 254, 263

Histogram analysis, 89, 91, 110

Histogram equalization, 31, 61, 89

Homogeneity, 222

Hough transform, generalized, 105, 111

HSI color space, 324, 333, 337

Hue, 324

I

IEEE1394'. *See* FireWire, 224

IIDC. *See* DCAM, 227

Illumination, 340

- back lighting, 16, 255
- bright field, 255
- dark field, 139, 256, 263
- flash, 142
- front lighting, 16
- on-axis, 263
- polarized, 122
- ring-shaped, 122, 248, 253, 257, 263, 281
- strobe, 263

Illumination gradient, 33, 37, 61, 107

Image addition. *See* Image arithmetic, addition, 34

Image analysis, 306

Image arithmetic, 25, 26, 37, 61

- addition, 34
- averaging, 34
- division, 37
- maximum, 37
- minimum, 37
- Shading correction, 38
- subtraction, 36

Image averaging. *See* Image arithmetic, averaging, 34

Image comparison, 5, 145

Image data transfer, 223
 Image distance, 243, 247, 249, 262, 242
 Image division. *See* Image arithmetic, division, 37
 Image preprocessing, 14, 17, 62, 86, 90, 91, 114, 120, 136, 139, 149, 150, 281
 Image rate, 207
 Image sensor. *See* Solid-state sensor, 204
 Image subtraction. *See* Image arithmetic, subtraction, 36
 Image transformation, 60
 Imaging equation, 239, 241, 242, 262
 Industrial ethernet, 341
 Information class, 342
 Infrared radiation, 248
 Ingress protection, 222
 Input layer, 164
 Input vector, 156–158, 163
 Inside bore holes, 257
 Instance-based classifier, 162, 168, 169, 171
 Integral transformations, 62
 Intensity, 324
 Interlaced scanning, 207
 Interpolation
 bilinear, 59
 pixel-repetition, 59
 Interpolation function, 108, 190

L

Labeling, 135
 Laplacian filter. *See* Filter, Laplacian, 51
 Laser line, 254, 292, 302
 Lateral illumination, 195, 258, 272
 Learning process, 166, 170
 Lens imaging, 249
 Lens speed, 238, 262
 Lens, 251, 253
 1", 2/3", 1/2", 1/3", 251
 C-mount, 249
 CS-mount, 249
 fisheye, 252
 size, 251, 263
 telecentric, 195, 201, 251, 258, 263
 ultra-wide-angle, 252
 zoom, 239, 246
 Light emitting diode, 253, 256–258, 263
 Light fields, 253
 Light guide, 254
 Lighting
 polarized, 256, 260, 264
 structured, 257, 263
 telecentric, 258, 264
 Line light guide, 256, 258

Light sensitivity, 142–144
 Linearity, 216, 222
 Line sensor, 204, 214
 Line symmetry, 72
 Line-scan camera. *See* Camera, line-scan, 231
 Loss of information, 10, 11

M

Macro image, 247
 Magnification, 178, 197, 239, 241, 262
 Maintenance, 341
 Manufacturing integration, 340
 Marginal ray, 241, 243, 251
 Mark identification, 5, 144
 Marking, 103, 121, 122, 129, 135, 150
 Mass data, 344
 Matching
 geometric, 106, 111
 shape-based, 104, 111
 template. *See* Template Matching, 95
 Matrix sensor, 204, 214
 Maximum image, 52
 Maximum operator, 37, 52
 Mean value filter. *See* Filter, mean value, 43
 Median filter. *See* Filter, median, 52
 Micro lens, 214
 Micrograph, 247
 Midspan devices, 231
 Minimum operator, 37
 Model circle, 192, 193
 Model Geometry, 200, 292
 Moments. *See* Object features, Moments, 314
 Multi-core processors, 344

N

Nearest neighbor classifier, 159, 171, 333
 Network, 342
 Neural networks, 170, 171
 NeuroCheck, 24
 Noise, 216
 amplifier, 222
 read.out, 222
 thermal, 222
 Notions of connectedness, 92

O

Object contour, 94, 304, 309, 312, 317, 318
 Object distance, 242, 247, 250, 262
 Object edge, 70, 82, 107, 111, 131

Object feature
 area, 282, 303–306, 317
 centroid, 304, 308, 309, 314, 317
 curvature, 312
 dimensions, 303
 enclosing rectangle, 304, 309, 317
 Euler number, 313, 318
 Feret's diameter, 309, 317
 fiber length, 312, 317
 fiber width, 313, 317
 form factor, 282, 307, 317
 Fourier descriptors, 314, 318
 gray level statistics, 315, 318
 moments, 314, 318
 origin, 303, 304
 perimeter, 282, 304, 306, 313, 317
 principal axis, 309, 317
 radii, 309, 317
 ratio, 304
 rectangularity, 304, 317
 texture, 315, 318
Object features, 318
Object group, 82
Object recognition, 5
Object search, 143
Object segmentation, 19
Object shape, 68, 309, 317
Object's center of gravity, 67, 72, 80, 82
Object's contour, 82, 84, 93
Opening, 56, 62
Optical character recognition (OCR), 14, 15, 18
Optical character recognition, 141
Orientation of an Object Group, 78
Origin. *See* Object features, origin, 303
Output amplifier, 207
Output layer, 164, 166, 168
Output unit, 168
Output value, 155, 165
Output vector, 155, 157, 171
Overlapped mode, 207

P

Parallelization, 7, 40, 144
Parameter, 343
Partial Scan, 219
Perceptron, 151, 170, 171
Performance specifications, 339
Perimeter. *See* Object features, perimeter, 305
Photon noise, 216
Pixel connectedness, 92
Pixel operation, 25, 29, 33, 61
Pixel pitch. *See* Pixel size, 213

Pixel repetition. *See* Interpolation, pixel repetition, 59
Pixel size, 213
Plug-and-Play, 226
Polarization filter, 122, 255–257, 260, 264
Polynomial classifier, 158, 163, 168, 171
Position adjustment, 68, 70, 82, 118, 135, 190, 192, 196, 268, 276, 281
Positioning, 5, 65, 84
 edge, 70
 entire object, 68
 object group, 79
 orientation primary axis, 73
 pivot, 79
 pseudo-shift, 80
 reference object, 68, 73, 75, 80, 82, 93, 147, 277, 282, 297
 reference, 68, 75, 133, 150, 192, 281, 282
 reference position, 66, 68, 78, 81, 82
 rotation contour, 75
 rotation individual object, 75
 rotation, 75, 78, 81, 118
 shift of individual object, 70
 shift, 75, 79–82
Positioning, 82
Position reference, 268
Power injectors, 231
Preprocessing. *See* Image preprocessing, 25
Presence verification, 65, 174, 302
Primary color, 212, 213
Principal axis method, 78, 81
Principal axis. *See* Object features, Principal axis, 309
Processing unit, 164, 166
Production line, 340
Programmable logic control, 7
Progressive scan, 207
Prototype pattern, 158, 161, 162
Pseudo-shift, 70, 80

Q

Quantum efficiency, 214, 215, 261

R

Radii. *See* Object features, Radii, 309
Radius, 191, 193, 287
Range thresholding, 90, 110
Rank filtering. *See* Filter, rank, 53
Ratio. *See* Object feature, Ratio, 304
RCE network, 161, 171
Real-time Ethernet, 341
Rectangular filter. *See* Filter, rectangular, 43

Rectangularity. *See* Object features, rectangularity, 304
 Reduction of information, 13
 Reference image, 26, 35–38, 145, 150
 Reference object. *See* Positioning, reference object, 82
 Reference part, 197, 199, 201, 301
 Reflection, 57, 122, 124, 138, 256, 257, 274
 Region of interest, 14
 Regression line, 185, 186, 189, 190, 192, 200, 292
 Relative response factor, 216
 Resampled image, 125, 143, 162, 167, 170, 285
 Resolution, 214
 Result, 343
 RGB color space, 324, 333, 337
 Ring illumination. *See* Illumination, ring-shaped, 122
 Ring light. *See* Illumination, ring-shaped, 122
 RLE. *See* Run length encoding, 305
 Robot positioning, 81
 Rolling shutter, 209
 Rotation, 70, 73, 98, 104, 110, 131, 133, 134
 Roundness, 193
 Run length encoding, 305

S

Sample parts, 340
 Sampling theorem, 10
 Saturation, 324
 Scaling method, 88
 Secondary axis, 72, 73
 Segmentation, 12, 14, 16, 20, 68, 111, 132, 134, 275, 284, 287, 289, 291, 297, 301
 binary, 92
 color, 319
 contour, 95
 template matching, 106
 Segmentation, 126
 Sensor element, 243, 262
 Serial interface, 342
 Shading correction. *See* Image arithmetic shading correction, 37
 Shift register. *See* Transport register, 206
 Shutter, 142
 Signal amplification, 218
 Signal-to-noise ratio, 221
 Smear, 216
 Sobel filter. *See* Filter, Sobel-, 51
 Sobel magnitude filter. *See* Filter, Sobel magnitude, 52

Solid-state sensor, 261
 color, 211, 261
 frame-transfer, 207
 full-frame, 207
 interline transfer, 206, 244
 multi-tap, 207, 210
 Solid-state sensors, 203
 Spatial discretization, 70, 306
 Special optical devices, 252
 Specification, 4, 133, 137, 146, 339
 Spectral sensitivity, 215, 261
 Square pixel, 176, 197, 200
 Standardization, 7
 Step size, 99
 Strobe lighting. *See* Lighting, strobe, 142
 Subpixel accuracy, 109, 200
 Subpixel computation. *See* Subpixel interpolation, 109
 Subpixel interpolation, 109, 196, 200

T

Telecentric lens. *See* Lens, telecentric, 195
 Template matching, 106, 110, 111, 132, 150, 162, 171, 268, 282, 284, 288, 301, 304, 316, 330
 coarse-to-fine-search, 100, 110, 133
 correlation points, 101
 optimizing, 99, 111
 principle, 96
 similarity measures, 96
 step size, 100, 104, 110, 133, 286
 subsampling, 100
 Textural features. *See* Object features, texture, 315
 Texture, 60
 Time-of-flight sensor, 210
 TOF sensor. *See* Time-of-flight sensor, 210
 Training, 127, 128, 149, 166, 167, 169
 Training data, 126, 127, 130, 137, 149, 155, 162, 166, 168, 170, 171
 Translation. *See* Positioning, shift, 80
 Transmitted light, 174, 191, 195, 199, 201, 255, 258, 263
 diffuse, 258, 264
 Transport register, 206
 Trigger, 142, 218

U

Unroll, 119, 238
 USB, 224, 228, 262

V

Variations in position, 65

Vector quantization, 161, 171

W

Wavelength, 215

Weighting factor, 165, 166, 171

White balance, 220

Width of object structures, 99, 140

Windows, 2, 344

X

Xenon lamps, 255

