Chapter 2
Simple Low Level Features for Image Analysis

Paolo Falcoz

Summary. As human beings, we perceive the world around us mainly through our eyes, and give what we see the status of "reality"; as such we historically tried to create ways of recording this reality so we could augment or extend our memory. From early attempts in photography like the image produced in 1826 by the French inventor Nicéphore Niépce (Figure 2.1) to the latest high definition camcorders, the number of recorded pieces of reality increased exponentially, posing the problem of managing all that information. Most of the raw video material produced today has lost its memory augmentation function, as it will hardly ever be viewed by any human; pervasive CCTVs are an example. They generate an enormous amount of data each day, but there is not enough "human processing power" to view them. Therefore the need for effective automatic image analysis tools is great, and a lot effort has been put in it, both from the academia and the industry. In this chapter, a review of some of the most important image analysis tools are presented.

Key words: Color Space, Blob detection, Shape Detection, Procrustes Analysis

2.1 Introduction

As human beings, we perceive the world around us mainly through our eyes, and give what we see the status of "reality"; as such we historically tried to create ways of recording this reality so we could augment or extend our memory. From early attempts in photography like the image produced in 1826 by the French inventor Nicéphore Niépce (Figure 2.1) to the latest high definition camcorders, the number of recorded pieces of reality increased exponentially, posing the problem of managing all that information. Most of the raw video material produced today has lost

Paolo Falcoz
University of Milan, Department of Information Technology, Via bramante 65, 26013 Crema Italy
e-mail: falcoz@dti.unimi.it

E. Damiani and J. Jeong (eds.), *Multimedia Techniques for Device and Ambient Intelligence*, DOI: 10.1007/978-0-387-88777-7_2,

its memory augmentation function, as it will hardly ever be viewed by any human; pervasive CCTVs are an example. They generate an enormous amount of data each day, but there is not enough "human processing power" to view them.

Therefore the need for effective automatic image analysis tools is great, and a lot effort has been put in it, both from the academia and the industry. Results from different research groups are impressive, and the DARPA[1] Grand and Urban Challenge [9] can be considered the showcase for the state-of-the-art in image processing. It may be useful to recall that the DARPA Grand Challenge is a prize competition for driverless cars, sponsored by the DARPA with the goal of developing technologies needed to create the first fully autonomous ground vehicle. The third event, The DARPA Urban Challenge, which took place on November 3, 2007, further advanced vehicle requirements to include autonomous operation in a mock urban environment. Robotics also has many meaningful examples of deep achievements in image processing, from the well known humanoid robot Asimo[14] to the less friendly machine-gun equipped sentry robot developed in South Korea by Korea University and Samsung [33]. More and more examples of complex image analysis tools embedded in consumer electronic equipments are available today, and the face detection feature built in some Canon cameras[2], and Sony camcorders[3] are an example.

Image processing algorithms can be extremely complex, but there are some basic operations and features that – whatever the complexity – are almost always considered. Among them we can mention:

- color analysis;
- edge extraction;
- shape matching;
- texture analysis.

For each of those features there exist many different algorithms with different goals and complexity, but taken alone most of them perform well only under specific conditions, and are lacking in the general case. Note that by "specific conditions" we include the need of having a dedicated training database, so that after training the algorithm works well only for the class of object for which it has been trained. A simple solution is to combine two or more different features together, so that the strengths of a feature can overcome the weaknesses of another, and vice versa. A similar problem has been faced by the MPEG-7 standard [21], which decided to make use of shape, region, and color descriptors altogether [2].

In the following sections we will discuss the meaning of "color" (section 2.2) and "color space" (sections 2.2.2 and 2.2.3); then we will use color to extract blobs (section 2.3). Different edge detectors will be presented in section 2.4, while in

[1] The **Defense Advanced Research Projects Agency** (DARPA) is an agency of the United States Department of Defense responsible for the development of new technology for use by the military. DARPA has been responsible for funding the development of many technologies which have had a major impact on the world, including **ARPANET**, the ancestor of the modern Internet.

[2] Canon Powershot S5 IS

[3] Sony HDR-CX12 HD AVCHD

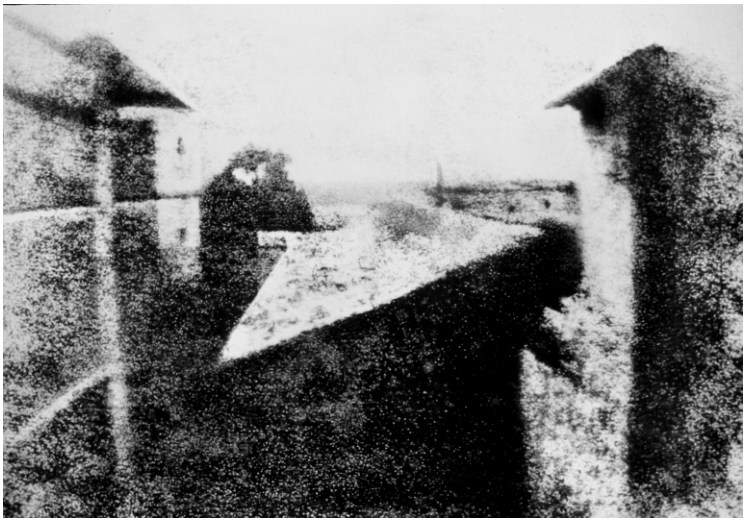


Fig. 2.1 Nicéphore Niépce's earliest surviving photograph, c. 1826 (View from the window of Le Gras). This image required an eight-hour exposure, which resulted in sunlight being visible on both sides of the buildings.

section 2.5.1 and 2.5.2 we will introduce **Procrustes Analysis** and **Iterative Closest Point** algorithm for shape registration (alignment).

Section 2.5.3 will deal with **Curvature Scale Space Descriptors** (CSSD), an effective way of describing shapes using scale, position, and rotation invariants; CSSD can be encoded for fast shape matching [28].

Section 2.6 presents some simple ideas for combining different features so that valuable knowledge can be extracted.

2.2 The Role of Color

From an anatomical point of view, all human interaction with "color" is mediated by the *retina*, the light-sensitive layer at the back of the eye that covers about 65 percent of its interior surface. Photosensitive cells called *rods* and *cones* in the retina convert incident light energy into signals that are carried to the brain by the optic nerve. Rods are attributed night vision, motion detection, and peripheral vision, while cones are attributed both color vision and the highest visual acuity [12]. In the middle of the retina is a small dimple called the *fovea* or *fovea centralis*. It is the center of the eye's sharpest vision and the location of most color perception. In fact, while cones are concentrated in the fovea, rods are absent there but dense elsewhere. Measured density curves for the rods and cones on the retina show an enormous density of cones in the fovea (Figure 2.2 (a)).

Considering for humans a global field of view of about 180° and a color field of view of about 15°, and translating the ratio between them into a 640×480 image, we find that actual color vision happens only within a 53×40 region. In Figure 2.2 (b) the inner rectangle is where color vision happens. If fact the rectangle should be blurred, because a small amount of cones is present also at bigger separation angles. Note that we used a rectangle only for simplicity, but a circle or an ellipse can be used as well.

From a perceptual point of view, "color" is the visual perceptual property corresponding in humans to the categories called red, yellow, blue, black, etc. Color categories and physical perception of color are obviously related with objects, materials, light sources, etc., and their physical properties of light absorption, reflection, and emission.

Color is therefore a very complex feature whose description depends on light characteristics, environment conditions, and sensor quality; the same "physical" red color with a wavelength of $780nm$ has different descriptions when perceived by a normal person, by a color-blind person, or by a webcam sensor (Figure 2.3).[4]

Despite its complexity and drawbacks, color is still a very important feature, used in many image processing tasks (ex. skin detector) with some clear advantages:

- it is very easy to compute;
- it is independent of image size and orientation.

However, in order to formalize the concept of color, we need to introduce the concept of *color space*.

2.2.1 Color Spaces

A *color model* is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components. When this model is associated with a precise description of how the components are to be interpreted (viewing conditions, etc.), the resulting set of colors is called a *color space*.

Adding a certain mapping function between the color model and a certain reference color space results in a definite "footprint" within the reference color space. This "footprint" is known as a gamut, and, in combination with the color model, defines a new color space. For example, Adobe RGB and sRGB are two different color spaces, both based on the RGB model.

The RGB color model is an additive color model in which red, green, and blue light are *added* together in various ways to reproduce a broad array of colors. It is additive in the sense that the three light beam are added together, and their light spectra add, wavelength for wavelength, to make the final color's spectrum. Zero intensity for each component gives the darkest color (no light, considered the black),

[4] Image taken from `http://en.wikipedia.org/wiki/Color_blindness`

a

b

Fig. 2.2 Original image (a), and proportion of the image actually seen in full color at any instant (b).

Fig. 2.3 An 1895 illustration of normal vision and various kinds of color blindness.

and full intensity of each gives a white. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

Oddly enough, the first known permanent color photo was taken by James Clerk Maxwell using the RGB color model developed by Thomas Young, Hermann Helmholtz and Maxwell himself. Figure 2.4 shows the photo, taken in 1861[5]. Those first experiments color photography involved the process of three color-filtered separate takes [13]. To reproduce the color photograph, three matching projections over a screen in a dark room were necessary.

Note that *subtractive* color models exist too; they work by partially or entirely masking certain colors on a typically white background (that is, absorbing particular wavelengths of light). Such models are called subtractive because colors "subtract"

[5] Image taken from Wikipedia, `http://en.wikipedia.org/wiki/RGB_color_model`

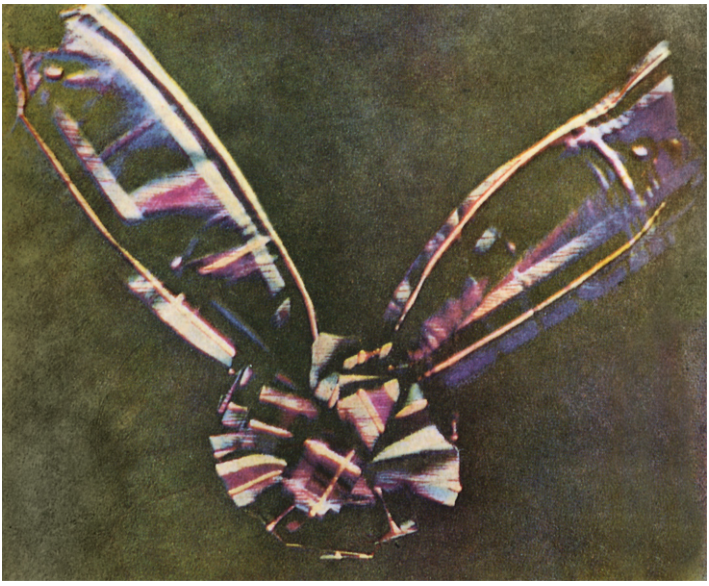


Fig. 2.4 The first permanent color photograph, taken by J. C. Maxwell in 1861 using three red, green, and violet-blue filters.

brightness from white. In the case of CMY those colors are cyan, magenta, and yellow.

There are many different color spaces, however when dealing wit human perception of color, only a few should be considered: those defined to be perceptually-uniform.

A perceptually-uniform color space is a color space in which any two colors that are perceived as "close" are "close" also in their numerical representation, and vice versa. For example, CIE-Lab is perceptually uniform, while Adobe RGB and sRGB are not.

In the next two subsections we will focus on three different color spaces, the first two – *HSL* and *HSV* – represent an attempt to derive a more perceptually uniform color space from RGB, while the third – *CIE-Lab* – was conceived to be perceptually uniform.

2.2.2 HSL and HSV

HSL and HSV are two related representations of points in an RGB color space, which attempt to describe perceptual color relationships more accurately than RGB, while remaining computationally simple. HSL stands for *hue*, *saturation*, *lightness*, while HSV stands for *hue*, *saturation*, *value*.

Both HSL and HSV describe colors as points in a cylinder (Figure 2.5) whose central axis ranges from black at the bottom to white at the top with neutral colors between them, where angle around the axis corresponds to "hue", distance from the axis corresponds to "saturation", and distance along the axis corresponds to "lightness", "value", or "brightness".

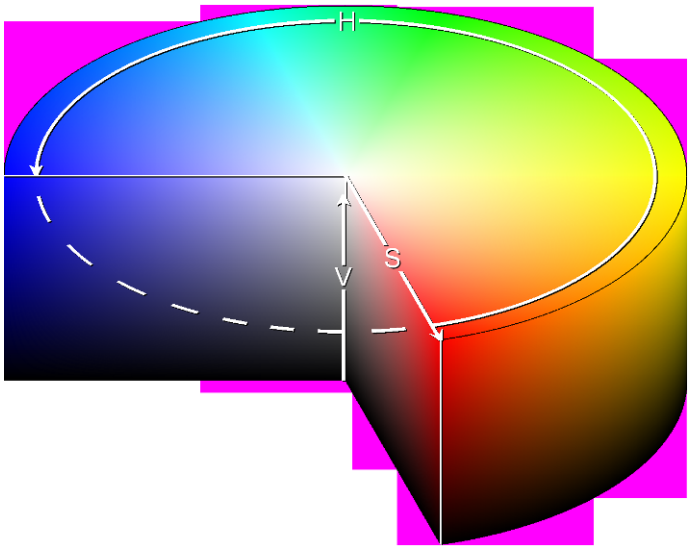


Fig. 2.5 Graphical representation of HSV cylinder

The two representations are similar in purpose, but differ somewhat in approach. Both are mathematically cylindrical, but while HSV (hue, saturation, value) can be thought of conceptually as an inverted cone of colors (with a black point at the bottom, and fully-saturated colors around a circle at the top), HSL conceptually represents a double-cone or sphere (with white at the top, black at the bottom, and the fully-saturated colors around the edge of a horizontal cross-section with middle gray at its center). Note that while "hue" in HSL and HSV refers to the same attribute, their definitions of "saturation" differ dramatically (Figure 2.6)[6].

Because HSL and HSV are simple transformations of RGB, the color defined by a (h,s,l) or (h,s,v) tuple depends on the particular color of red, green, and blue "primaries" used. Note that in practice those primaries are strictly related to the technology used to generate them; the actual "blue" color generated by the blue electron gun used in cathode ray devices is different from the blue generated by the blue LEDs of LED devices, and is different from the blue detectors of a CCD camera. Each unique RGB device therefore has unique HSL and HSV spaces to accompany it. An (h,s,l) or (h,s,v) tuple can however become definite when it is tied to a particular RGB color space, such as sRGB.

[6] Image taken from `http://en.wikipedia.org/wiki/HSL_and_HSV`

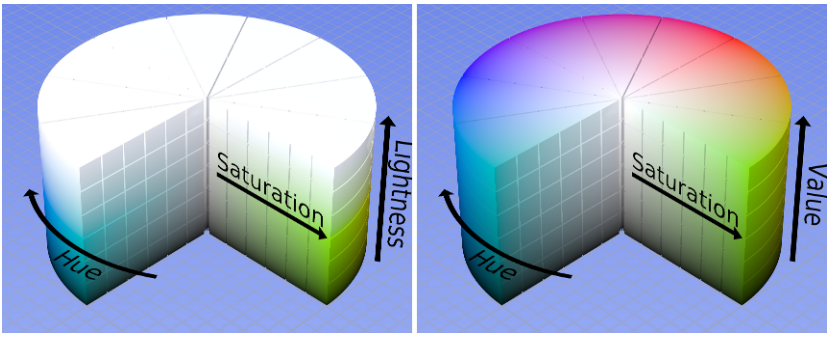


Fig. 2.6 Comparison of the HSL and HSV color spaces.

Both models were first formally described in 1978 by Alvy Ray Smith [30], though the concept of describing colors by these three dimensions, or equivalents such as hue, chroma, and tint, was introduced much earlier [27].

2.2.3 CIE-Lab

CIELAB is the second of two systems adopted by CIE[7] in 1976 as models that better showed uniform color spacing in their values. CIELAB is an opponent color system based on the earlier (1942) system of Richard Hunter [17][18] called *L*, *a*, *b*. Color opposition correlates with discoveries in the mid-1960s that somewhere between the optical nerve and the brain, retinal color stimuli are translated into distinctions between light and dark, red and green, and blue and yellow. CIELAB indicates these values with three axes: L*, a*, and b* (Figure 2.7)[8].

The central vertical axis represents lightness (signified as L*) whose values run from 0 (black) to 100 (white). This scale is closely related to Munsell's [25][26] value axis except that the value of each step is much greater. This is the same lightness valuation used in CIELUV.

The color axes are based on the fact that a color can't be both red and green, or both blue and yellow, because these colors oppose each other. On each axis the values run from positive to negative. On the a-a' axis, positive values indicate amounts of red while negative values indicate amounts of green. On the b-b' axis, yellow is positive and blue is negative. For both axes, zero is neutral gray.

Therefore, values are only needed for two color axes and for the lightness or grayscale axis (L*), which is separate (unlike in RGB, CMY or XYZ where lightness depends on relative amounts of the three color channels).

[7] Comission Internationale de l'Eclairage

[8] The full nomenclature is 1976 CIE L*a*b* Space.

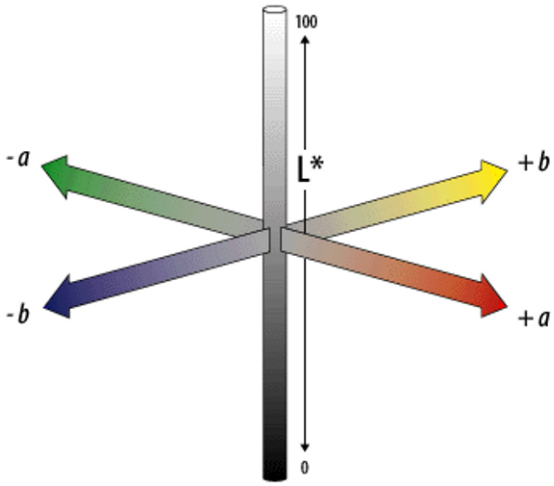


Fig. 2.7 Graphical representation of CIE-Lab color space

2.2.4 Color Flattening

Now that we know what color is from an abstract point of view, we are ready to work with actual colors from digital images and videos. The problem is that they are mostly shot with low cost, low quality equipment, meaning non uniform colors and evident noise.

One method to cope with this is to flatten colors: a very common filter used for this purpose is the *blur* filter.

The problem with this approach is that not only noise but also edges are flattened, causing lost of potentially important information. A better solution is to perform several steps of bilateral filtering [34] (Figure 2.8).

The effectiveness of this approach is to combine a low-pass filter with a range filter

$$h(x) = k^{-1}(x) \int_{-\inf}^{\inf} \int_{-\inf}^{\inf} f(\xi) c(\xi, x) s(f(\xi), f(x)) \mathrm{d}\xi$$

where

$$k(x) = \int_{-\inf}^{\inf} \int_{-\inf}^{\inf} c(\xi, x) s(f(\xi), f(x)) \mathrm{d}\xi$$

$c(\xi, x)$ measures the geometric closeness between the neighborhood center x and a nearby point ξ, $s(f(\xi), f(x))$ measures the photometric similarity between the pixel at the neighborhood center x and that of a nearby point ξ, and $f()$ represents the image function. The low-pass filter is defined by $c(\xi, x)$, while the range filter

is defined by $s(f(\xi), f(x))$. Since this technique combines two different filters, it is called *bilateral filtering*.

The actual implementation of low-pass and range filters can be based on simple Gaussian filtering, where both the closeness function $c(\xi, x)$ and the similarity function $s(f(\xi), f(x))$ are Gaussian functions of the Euclidean distance between their arguments. Closeness then becomes

$$c(\xi, x) = e^{-\frac{1}{2}\left(\frac{d(\xi, x)}{\sigma_d}\right)^2}$$

where

$$d(\xi, x) = d(\xi - x) = |\xi - x|$$

while similarity becomes

$$s(\xi, x) = e^{-\frac{1}{2}\left(\frac{\delta(f(\xi), f(x))}{\sigma_r}\right)^2}$$

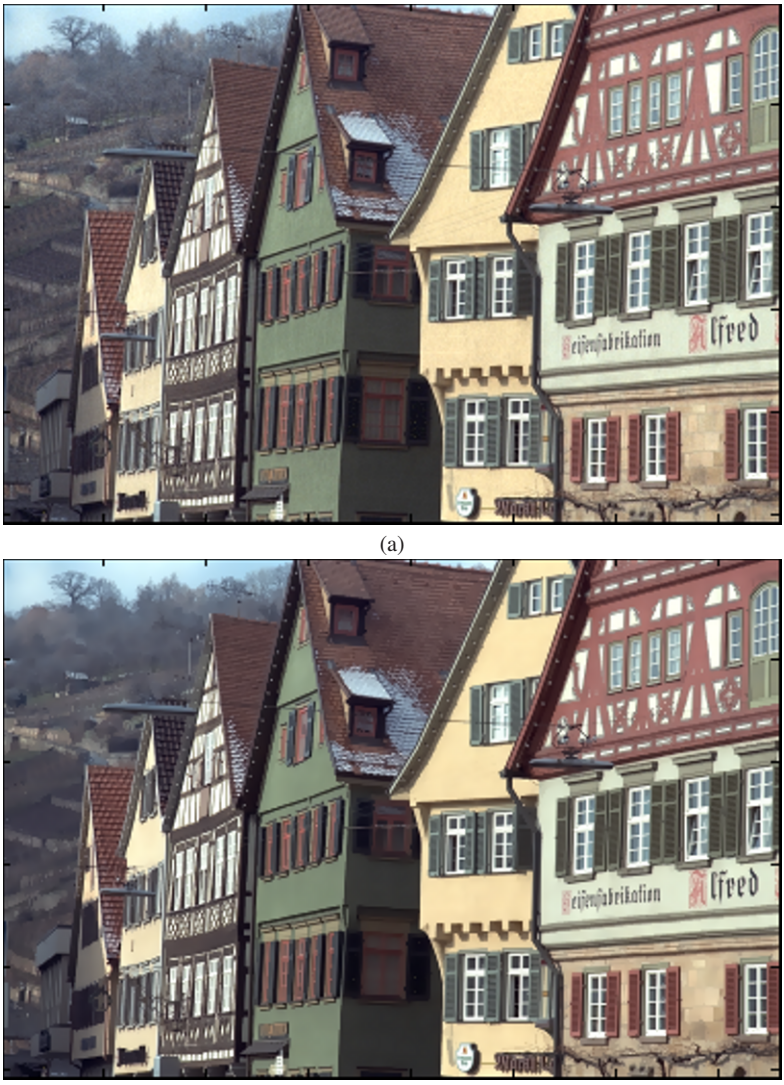
where

$$\delta(\phi, f) = \delta(\phi - f) = |\phi - f|$$

The meaning of bilateral filtering is to replace the pixel value at x with an average of similar (photometric similarity) and nearby (geometric closeness) pixel values. In smooth regions, pixel values in a small neighborhood are similar to each other, and the normalized similarity function is close to one. As a consequence, the bilateral filter acts essentially as a standard domain filter, and averages away the small, weakly correlated differences between pixel values caused by noise. Consider now a sharp boundary between a dark and a bright region. Suppose on the other hand that the bilateral filter is centered, on a pixel on the bright side of the boundary, then the similarity function assumes values close to one for pixels on the same side, and close to zero for pixels on the dark side. The normalization term x ensures that the weights for all the pixels add up to one. As a result, the filter replaces the bright pixel at the center by an average of the bright pixels in its vicinity, and essentially ignores the dark pixels. Conversely, when the filter is centered on a dark pixel, the bright pixels are ignored instead. Thus, good filtering behavior is achieved at the boundaries, thanks to the domain component of the filter, and crisp edges are preserved at the same time, thanks to the range component.

2.3 Blob Detection

Blob detection and extraction proves to be a useful tool in many areas; one main application is to provide complementary information about regions, which is not obtained from edge detectors or corner detectors. In early work in the area, blob detection was used to obtain regions of interest for further processing. These re-



(a)

(b)

Fig. 2.8 Original (a) and corresponding flattened (b) image (4 steps)

gions could signal the presence of objects or parts of objects in the image domain with application to object recognition and/or object tracking. In other domains, such as histogram analysis, blob descriptors can also be used for peak detection with application to segmentation. Another common use of blob descriptors is as main primitives for texture analysis and texture recognition. In more recent work, blob descriptors have found increasingly popular use as interest points for wide baseline

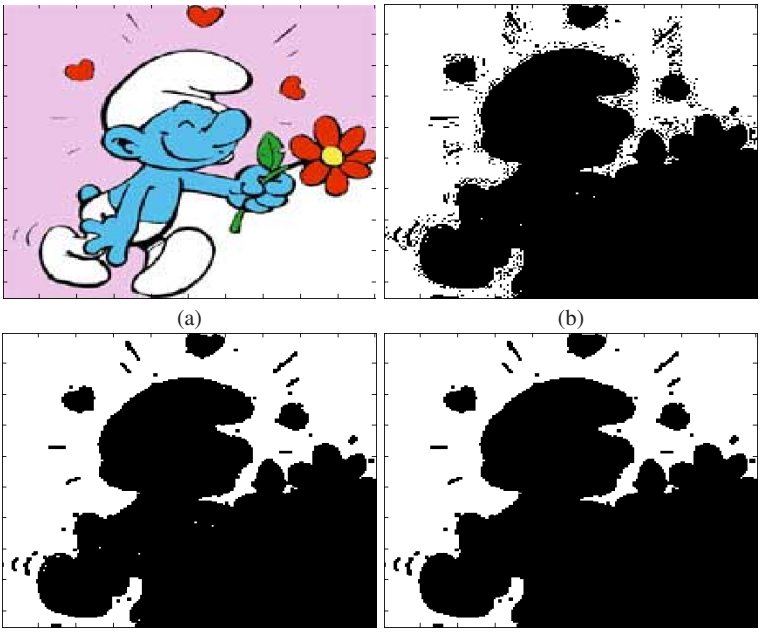


Fig. 2.9 Original image (a), mask (white) of lilac blob (b), mask after morphological closing (c), mask after noise removal (d).

stereo matching [22] and to signal the presence of informative image features for appearance-based object recognition based on local image statistics.

Simple blob extraction and refinement based on color ranges can be achieved using the following idea:

1. given the color input image I, create a binary matrix M with the same width and height of I, and set all the elements to 0;
2. scan the input image element by element and check if the value of each color plane falls within the specified range. If yes, then put the corresponding mask's element to 1 (Figure 2.9 (b));
3. apply a morphological "closing" (dilation followed by erosion) to M in order to fill holes and to smooth blobs (Figure 2.9 (c));
4. remove isolated group of pixels smaller than a given threshold (Figure 2.9 (d));
5. scan M and label all 8-connected blobs. Labeling can be done using the technique outlined in [10]. Each blob's mass is then calculated, along with color statistics.

Note that the previous procedure can be applied to blob detection based on characteristics other than color, the only changing part is the one that assigns an element to the blob or to the background (the non-blob area)[15]. Since there can be many blobs of the same color (with the same characteristic), a selection criterion can be

used in order to take only the n best ones; foe example, if "best" means "biggest", then only the biggest n blobs are considered.

2.4 Edge Detection

From an algorithmic point of view, *edge detection* translates into detecting sharp changes in image brightness; the underlying assumption is that such brightness changes are strongly correlated with important events and properties changes of the world (represented by the image). In general, discontinuities in image brightness are likely to correspond to:

- discontinuities in depth;
- discontinuities in surface orientation;
- changes in material properties;
- variations in scene illumination.

In the ideal case, the result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well curves that correspond to discontinuities in surface orientation. Thus, applying an edge detector to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified. Unfortunately, however, it is not always possible to obtain such ideal edges from real life images of moderate complexity. Edges extracted from non-trivial images are often hampered by fragmentation, meaning that the edge curves are not connected, missing edge segments as well as false edges not corresponding to interesting phenomena in the image – thus complicating the subsequent task of interpreting the image data.

There are many different algorithms for computing edges [32][29][11][35], but three must be cited:

- Prewitt operator [31];
- Sobel operator;
- Canny filter [3].

The best of the three is the Canny filter, the other two are interesting because they are simple and fast (Figure 2.10).

Mathematically, both the Sobel and Prewitt operators use two 3x3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. Given the input image I, the output images $\mathbf{G_x}$ and $\mathbf{G_y}$ which at each point contain the horizontal and vertical derivative approximations, are calculated as follows

$$G_x^S = S_v \otimes I \qquad G_y^S = S_h \otimes I$$
$$G_x^P = P_v \otimes I \qquad G_y^P = P_h \otimes I$$

where $\otimes$ denotes the bidimensional convolution operator, and

$$S_v = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad S_h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$
$$P_v = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad P_h = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

denotes the kernels for vertical and horizontal changes of the Sobel and Prewitt operators.

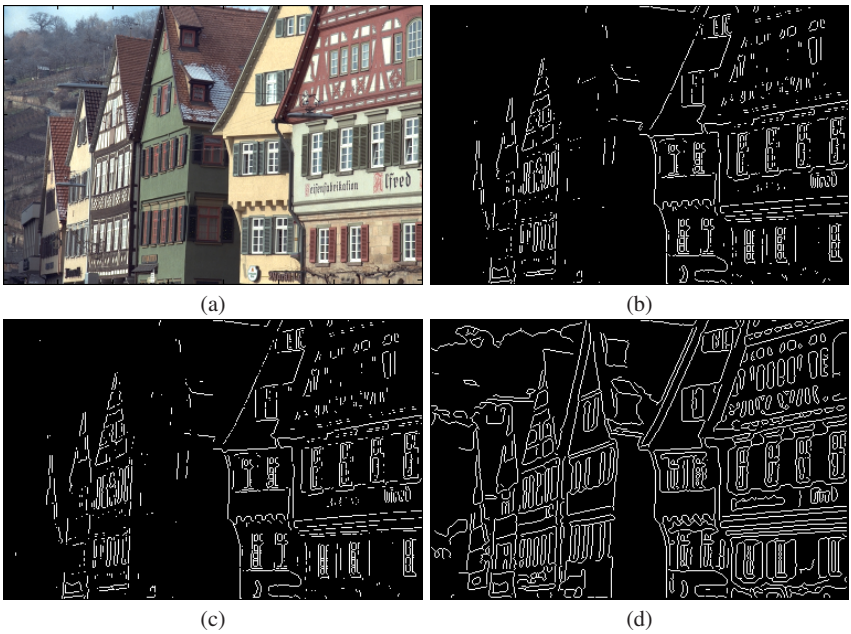


Fig. 2.10 Original image (a), Sobel (b), Prewitt (c), and Canny (d) edge detectors.

Canny builds on top of Sobel/Prewitt operators, considering the mathematical problem of deriving an optimal smoothing filter given the criteria of detection, localization and minimizing multiple responses to a single edge. He showed that the optimal filter given these assumptions is a sum of four exponential terms. He also showed that this filter can be well approximated by first-order derivatives of Gaus-

sians. Canny also introduced the notion of non-maximum suppression, which means that the image is scanned along the image gradient direction, and if pixels are not part of the local maxima they are set to zero. This has the effect of supressing all image information that is not part of local maxima.

Because the Canny edge detector uses a filter based on the first derivative of a Gaussian, it is susceptible to noise present on raw unprocessed image data, so the first step is to convolve the raw image with a Gaussian filter. The result is as a slightly blurred version of the original which is not affected by a single noisy pixel to any significant degree.

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator (Prewitt, Sobel for example) returns a value for the first derivative in the horizontal direction ($\mathbf{G_y}$) and the vertical direction ($\mathbf{G_x}$). From this the edge gradient and direction can be determined:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

$$\Theta = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).

Given estimates of the image gradients, a search is then carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction (non maximum suppression). So, for example, if the rounded angle is zero degrees the point will be considered to be on the edge if its intensity is greater than the intensities in the north and south directions, if the rounded angle is 90 degrees the point will be considered to be on the edge if its intensity is greater than the intensities in the east and west directions, if the rounded angle is 135 degrees the point will be considered to be on the edge if its intensity is greater than the intensities in the north east and south west directions, if the rounded angle is 45 degrees the point will be considered to be on the edge if its intensity is greater than the intensities in the south east and north west directions. This is worked out by passing a 3x3 grid over the intensity map.

From this stage a set of edge points, in the form of a binary image, is obtained.

Intensity gradients which are large are more likely to correspond to edges than if they are small. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis.

Thresholding with hysteresis requires two thresholds – high and low. Making the assumption that important edges should be along continuous curves in the image allows us to follow a faint section of a given line and to discard a few noisy pixels that do not constitute a line but have produced large gradients. Therefore we begin by applying a high threshold. This marks out the edges we can be fairly sure are genuine. Starting from these, using the directional information derived earlier,

edges can be traced through the image. While tracing an edge, we apply the lower threshold, allowing us to trace faint sections of edges as long as we find a starting point.

Once this process is complete we have a binary image where each pixel is marked as either an edge pixel or a non-edge pixel.

2.5 Simple Shapes

Edge detectors are a fundamental step in shape analysis, as well as algorithms for shape comparison and matching. Shape matching usuallly requires the evaluation of a distance between the shape themselves or their projection in some other feature space; n-dimensional (or vector) euclidean distance is a good candidate.

2.5.1 Scale and Position Invariants: Procrustes Analysis

In order to compare two shapes we need to make them independent of position and size.

Procrustes analysis is a form of statistical shape analysis used to analyse the distribution of a set of shapes. The name Procrustes refers to a bandit from Greek mythology who made his victims fit his bed either by stretching their limbs or cutting them off.

Here we just consider objects made up from a finite number k of points in n dimensions; these points are called *landmark points*.

The shape of object can be considered as a member of an equivalence class formed by removing the translational, rotational and scaling components.

For example, translational components can be removed from an object by translating the object so that the mean of all the points lies at the origin. Likewise the scale component can be removed by scaling the object so that the sum of the squared distances from the points to the origin is 1.

Mathematically: take k points in two dimensions,

$$((x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k))$$

The mean of these points is $(\bar{x}, \bar{y})$, where

$$\bar{x} = \frac{1}{k} \sum_{i=1}^{k} x_i \qquad \bar{y} = \frac{1}{k} \sum_{i=1}^{k} y_i$$

Now translate these points so that the mean is translated to the origin $(x, y) \rightarrow (x - \bar{x}, y - \bar{y})$, giving the point $(x_1 - \bar{x}, y_1 - \bar{y}), \ldots$. Likewise scale can be removed by finding the size of the object

$$s = \sqrt{(x_1 - \bar{x})^2 + (y_1 - \bar{y})^2 + \cdots}$$

and dividing the points by the scale giving points $((x_1 - \bar{x})/s, (y_1 - \bar{y})/s)$. Other methods for removing the scale can also be used.

2.5.2 Shape Alignment: Iterative Closest Point

Iterative Closest Point (ICP) was introduced by Besl and McKay in 1992 [1] and solves the general problem of matching two clouds of points.

This matching technique can be used from simple 2D shape alignment to complex 3D surfaces reconstruction. The algorithm is very simple and is commonly used in real-time.

The goal of ICP is to find the rigid transformation T that best aligns a cloud of scene points S with a geometric model M. The alignment process works to minimize the mean squared distance between scene points and their closest model point. ICP is efficient, with average case complexity of $O(n \log n)$ for n point images and it converges monotonically to a local minimum. At each iteration, the algorithm computes correspondences by finding closest points and, then, minimizes the mean square error in position between the correspondences [6] [16]. A good initial estimate of the transformation is required and all scene points are assumed to have correspondences in the model[9].

Algorithm 1: Iterative Closest Point

Initial situation: Let S be a set of N_s points $\{s_1, \ldots, s_{N_s}\}$, and let M be the model. Let $\|\mathbf{s} - \mathbf{m}\|$ be the distance between points $\mathbf{s} \in S$ and $\mathbf{m} \in M$, and let $CP(s_i, M)$ be the closest point in M to the scene point s_i;
Phase 1: Let T_0 be an initial estimate of the transformation;
Phase 2: Repeat for $k = 1, \ldots, k_{max}$ or until ternination criteria is met

1. Build the set of corrispondences

$$C = \bigcup_{i=1}^{N_s} \{(T_{k-1}(s_i), CP(T_{k-1}(s_i), M))\}$$

2. Compute the new transformation T_k that minimizes mean square error between point pairs in C [6] [16]

The result will be the refined transformation $T_{k_{max}}$(translation, rotation).

For rigid deformation, the distance used in ICP is only the Euclidean distance. Point in rigid deformation is 3D point with components (x, y, z) [1][36].

[9] If the model shape can be parameterized, this limitation can be overcome by generating a model with a number of points equal to that of the scene.

For non-rigid deformation, it is not correct anymore to say that corresponding points have the closest Euclidean distance. The distance should be redefined to describe the similarity between corresponding points. Feldmar [7] defines a 3D Euclidean point in 8D with normal (n_x, n_y, n_z) and principal (k_1, k_2) curvatures in addition to the (x, y, z) components. Given two surfaces S_1 and S_2, his definition is:

$$d(M,N) = (\alpha_1(x-x')^2 + \alpha_2(y-y')^2 + \alpha_3(z-z')^2 + \alpha_4(n_x-n'_x)^2 + \alpha_5(n_y-n'_y)^2 + \alpha_6(n_z-n'_z)^2 + \alpha_7(k_1-k'_1)^2 + \alpha_8(k_2-k'_2)^2)^{1/2}$$

where M is a point on surface S_1, N is a point on surface S_2, (n_x, n_y, n_z) is the normal on S_1 at point M, k_1, k_2 are the principal curvatures, and α_i is the difference between the maximal and minimal value of the i^{th} coordinate of points in S_2.

In his definition both global and local affine are implemented.

2.5.3 Shape Encoding and Matching: Curvature Space Scale

The curvature scale space (CSS) was introduced by Mokhtarian and Mackworth [23] [24] as a shape representation for planar curves.

The representation is computed by convolving a path-based representation of the curve with a Gaussian function, as the standard deviation of the Gaussian varies from a small to a large value, and extracting the curvature zero-crossing points of the resulting curves. The representation is essentially invariant under rotation, uniform scaling, and translation of the curve. This and a number of other properties makes it suitable for recognizing a noisy curve of arbitrary shape at any scale or orientation. After substantial and comprehensive testing, the CSS technique was selected as a contour shape descriptor for MPEG-7 [21].

To create a CSS description of a contour shape, N equi-distant points are selected on the contour, starting from an arbitrary point on the contour and following the contour clockwise. The x and $y$coordinates of the selected N points are grouped together into two series X and Y. The contour is then gradually smoothed by repetitive application of a Gaussian kernel[10] to X and Y coordinates of the selected contour points. As a result of the smoothing, the contour evolves and its concave parts gradually flatten-out, until the contour becomes convex. A so-called CSS image can be associated with the contour evolution process[11].

The CSS image horizontal coordinates correspond to the indices of the contour points selected to represent the contour $(1, \ldots, N)$, and CSS image vertical coordinates correspond to the amount of filtering applied, defined as the number of passes of the filter. Each horizontal line in the CSS image corresponds to the smoothed contour resulting from k passes of the filter (Figure 2.11). For each smoothed contour,

[10] The MPEG-7 standard uses a low-pass filter with the kernel (0.25, 0.5, 0.25)

[11] the CSS image does not have to be explicitly extracted, but is useful to illustrate the CSS representation.

the zero-crossings of its curvature function are computed. Curvature zero-crossing points separate concave and convex parts of the contour. The CSS image has characteristic peaks. The coordinate values of the prominent peaks (x_{css}, y_{css}) in the CSS image are extracted; in addition, the eccentricity and circularity of the contour can also be calculated.

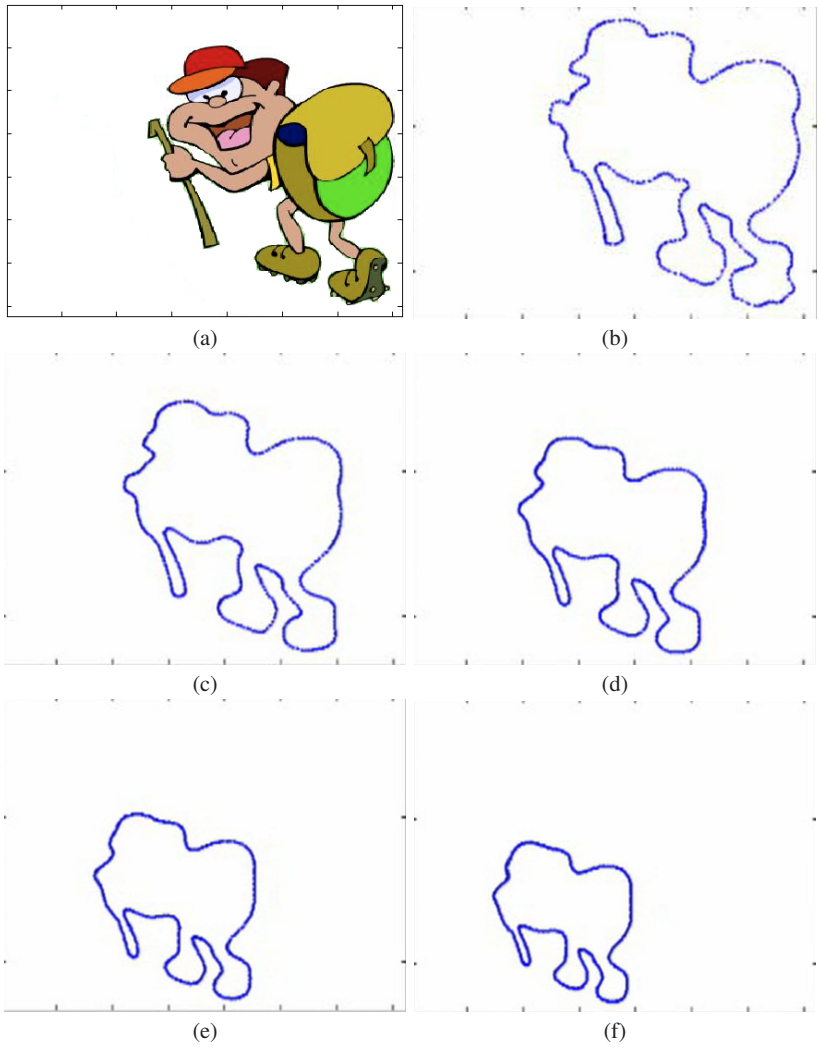


Fig. 2.11 Original image (a), outer contour (b), after 15 filtering steps (c), 30 filtering steps (d), after 45 filtering steps (e), after 60 filtering steps (f).

Once the css descriptors have been extracted, the shape is ready for matching. The canonical CSS (CCSS) based shape retrieval algorithm [5] is to compare a CSS

descriptor of query image with a set of CSS descriptor of database images, and to return the n best match as its output. In order to find the minimum cost of the match between a query image and a database image, the algorithm must consider all possible ways of aligning the contour maxima from both CSS images, and compute the associated cost by shifting the query CSS image or a database CSS image.

Unfortunately, in general the computation of a CSS image can take a long time, making it difficult to apply the method to real-time object recognition.

To overcome this limitation, several variations and hybridizations of the original algorithm have been proposed [19] [28] [37].

2.6 Combination of simple features

We will now use some of the features presented in the previous sections to build a sky detector; there exist many good sky detectors, but our aim is to show that with the combination of a few simple generic features, interesting results can be obtained. This is just an example and from the point of view of pure performance it cannot be compared to dedicated algorithms [20] [8].

The idea behind our sky detector is the following:

1. extract blue blobs B from input image I;
2. use a simple texture analysis to discriminate between sky blobs and non-sky blobs;
3. extract edges E from I using Canny edge detector;
4. perform binary *and* between B and E to generate the combination mask C;
5. perform morphological closing over C;
6. run again texture analysis to discriminate between sky blobs and non-sky blobs.

The first step is to define what is the meaning of "sky" from the point of view of the color "blue": after a manual sampling over some twenty images with sky, we define "blue" to be the color in the following HSV range

$$\begin{cases} 140 \leq x \leq 300 & x \in H,\ H = \{h \in \mathscr{R}, 0 \leq h \leq 360\} \\ 0 \leq y \leq 0.45 & y \in S,\ S = \{s \in \mathscr{R}, 0 \leq s \leq 1\} \\ 76 \leq z \leq 255 & z \in V,\ V = \{v \in \mathscr{N}, 0 \leq v \leq 255\} \end{cases}$$

The result of blob extraction using this definition can be seen in Figure 2.13 (b). There are many non-sky blobs, so we use a simple consideration made by Luo [20] to discriminate good blobs from bad blobs: as a result of the physics of light scattering by small particles in the air, clear sky often appears in the shade of deep, saturated blue at the top of the image and gradually desaturates to almost white towards a distant horizon line in the image.

What we need is then a gradient detector to measure this desaturation effect. The simplest way we can think of, is to measure the difference in saturation between the top and the bottom pixels of each blobs; if the difference is bigger than a threshold, then we mark the blob as sky, otherwise as non-sky. Many improvements can be

done, but even in its naive simplicity this approach works well enouh for our purpose (Figure 2.13 (c)).

The effect of texture analysis is to delete many but not all non-sky blobs, so the next step is to take advantage of edge detection to better partition sky blobs. It is evident from Figure 2.13 (c) that the big sky blob is the sum of a little "true" sky blob at the top plus a portion of a hill at the bottom. From Figure 2.12 (a) we can see that the edge detector detects the border line between the sky and the hill, so we superimpose the edges to the blob and check if there are edges that partition the blob (Figure 2.12 (d)). To make those "fractures" more evident we perform a morphological closing (Figure 2.12 (e)).

The new blobs created are visible in Figure 2.13 (d).

The last step is to re-run texture analysis and discriminate again good from bad blobs (Figure 2.13 (e), (f)). The result is that only the true sky blob is marked as good.

In many cases this simple algorithm works well, however it depends much on the quality of the edges found, and tends to over-fragment the blobs.

The same ideas can be used to find vegetation, skin, and so on by changing the color definition and the texture discrimination function; constraints on blobs shape can be added using the algorithms presented in Section 2.5.

2.7 Conclusions

The goal of this chapter was to introduce some basic ideas on image features extraction, particularly from the point of view of color, edges, and shapes. The algorithms presented are well-known and widely used, and even if some of them were conceived many years ago, they still can be considered state-of-the-art.

In the last section we used an example to introduce some simple yet useful hybridization ideas, showing how to combine different techniques to extract the sky blobs from an image. This a the first step in image understanding: knowing if there is sky or not can lead to considerations on the environment (indoor/outdoor), and the weather (color of the sky, fragmentation of the blobs due to clouds). Even if we are not interested in the presence of sky, this first step can be used to delete from the image useless blobs, and therefore reduce the search space.

The combination of two or more of such simple detectors, can generate not trivial context information which can be used in turn to make higher order logical reasoning; those higher order informations then will be the new features, ready to be combined again and generate deeper image comprehension.

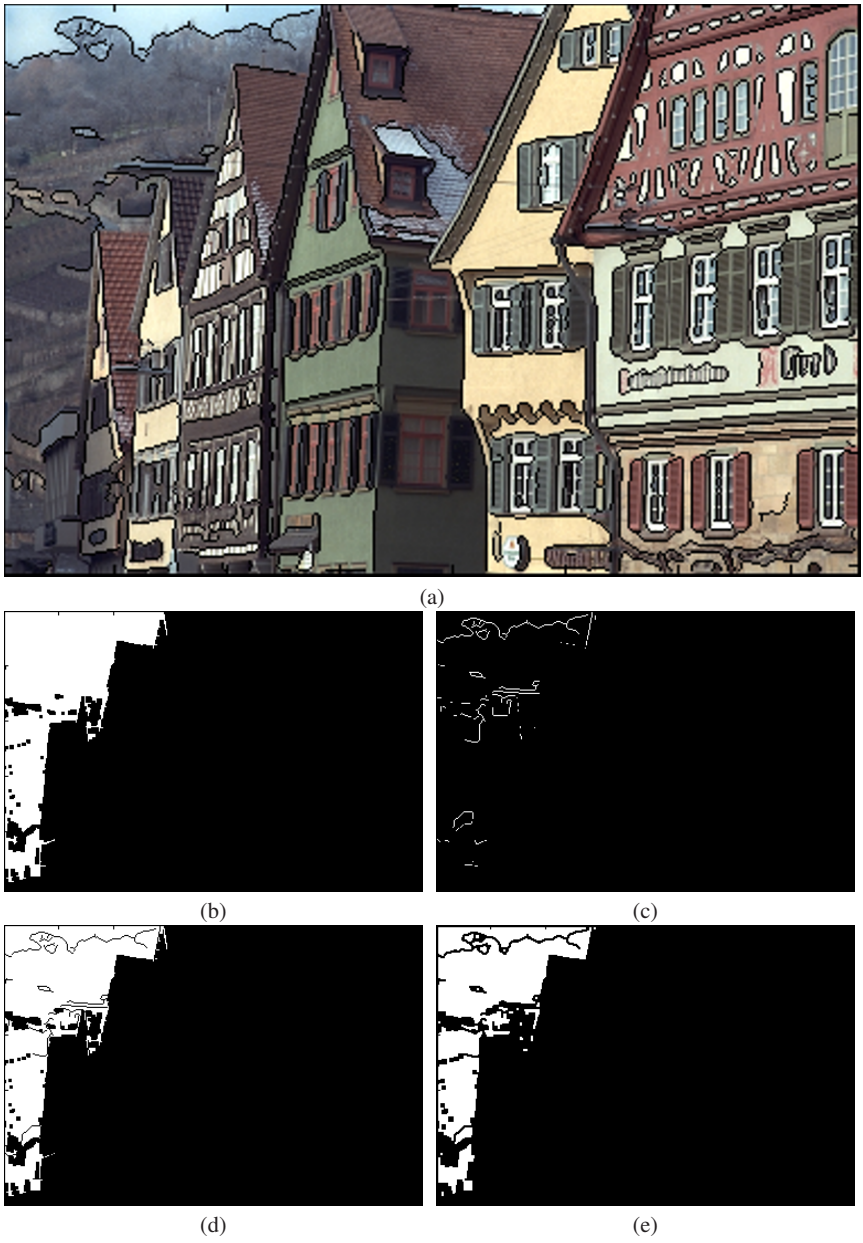


Fig. 2.12 Superimposition of original image and Canny edge detector (a), original blob (b), edges within the blob (c), blob partition according to edges (d), partition after morphological closing (e).

References

1. Besl,P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. on Pattern Analysis and Machine Intelligence, **14**, 239–256 (1992)

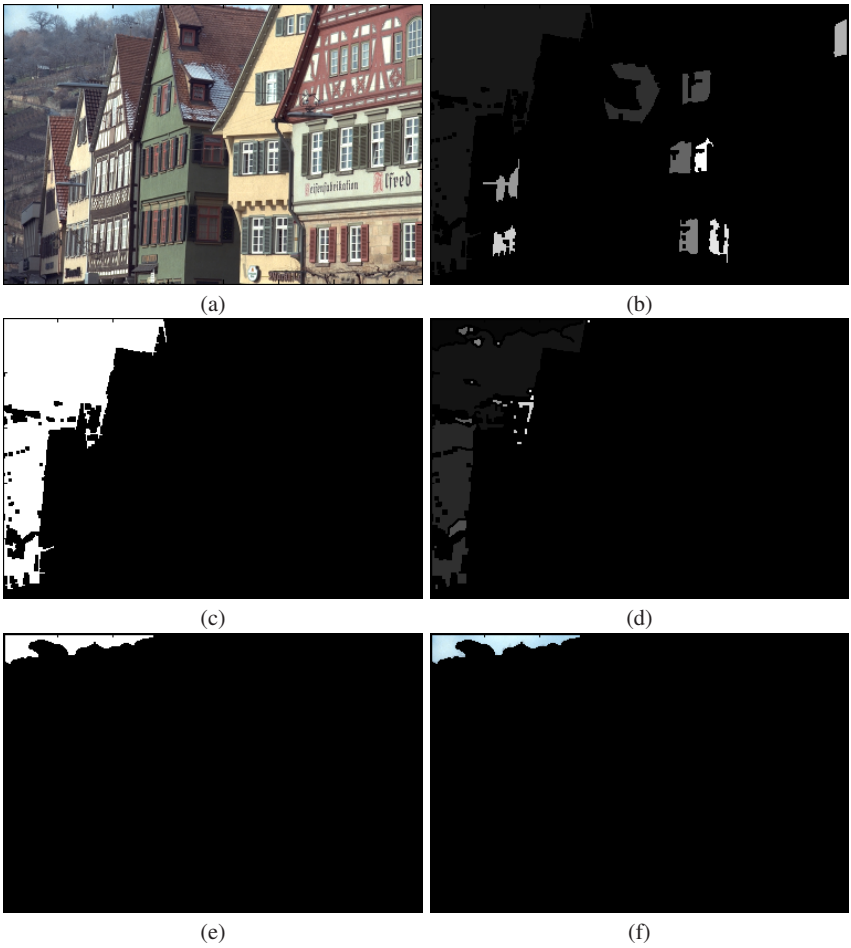


Fig. 2.13 Original image (a), after color blob extraction (b), after texture filtering (c), after edge combination (d), after second texture filtering (e), final sky blob (f).

2. M. Bover: MPEG-7 Visual Shape Descriptors. IEEE Trans.s on Circuits and Systems for Video Technology, **11**, 716–719 (2001)
3. J. Canny: A Computational Approach To Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence,**8**, 679–714 (1986)
4. Chen, Y., Sundaram, H.: Estimating Complexity of 2D Shapes, IEEE 7th Workshop on Multimedia Signal Processing, pp. 1-4, Shanghai (2005)
5. Cheng, H.D., Li, J.: Fuzzy homogeneity and scale-space approach to color image segmentation, Pattern Recognition, **36**, 1545–1562 (2003)
6. Faugeras, O. D., Hebert, M.: The Representation, Recognition, and Locating of 3-D Objects. Int. J. Robotics Research, **5(3)**, 27–52 (1986)
7. Feldmar, J., Ayache, N. J.: Rigid, affine and locally affine registration of free-form surfaces, Int. J. on Computer Vision, **18**, 99–119 (1996)
8. Gallagher, A. C., Luo, J., Hao,W.: Improved Blue Sky Detection Using Polynomial Model Fit. ICIP, **4**, 2367–2370 (2004)

9. DARPA: http://www.darpa.mil/GRANDCHALLENGE/
10. Haralick, R.M., Shapiro, L.G.: Computer and Robot Vision. Vol. 1, Addison-Wesley, pp. 28-48, (1992)
11. Harris, C., Stephens, M.: A combined corner and edge detector. Proc. of the 4^{th} Alvey Vision Conference, pp 147–151 (1988)
12. E. Hecht, Optics, 2^{nd} Edition. Addison Wesley (1987)
13. R. Hirsch: Exploring Colour Photography: A Complete Guide. Laurence King Publishing (2004)
14. Honda: http://world.honda.com/ASIMO/
15. Horn, B. K. P.: Robot Vision, pp. 69–71, MIT Press (1986)
16. Horn, B. K. P.: Closed Form Solutions of Absolute Orientation Using Unit Quaternions. Journal of the Optical Society of America, **4**(4), 629–642 (1987)
17. Hunter, R. S.: Photoelectric Color-Difference Meter. Proceedings of the Winter Meeting of the Optical Society of America (1948)
18. Hunter, R. S.: Accuracy, Precision, and Stability of New Photo-electric Color-Difference Meter. Proc. of the Thirty-Third Annual Meeting of the Optical Society of America, (1948)
19. Kopf, S., Haenselmann T., Effelsberg, W.: Enhancing Curvature Scale Space Features for Robust Shape Classification. Proc. of International Conference on Multimedia & Expo (2005)
20. Luo, J., P. Etz, S.: A physical model-based approach to detecting sky in photographic Images. IEEE Trans. on Image Processing, **11**, 201–212 (2002)
21. MPEG-7 Standard: http://www.chiariglione.org/MPEG/standards/mpeg-7/mpeg-7.htm
22. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremum regions. Proc. of British Machine Vision Conference, pp. 384–393 (2002)
23. Mokhtarian, F., Mackworth, A.K.: Scale-based description and recognition of planar curves and two-dimensional shapes. IEEE Trans. Pattern Analysis and Machine Intelligence, **8**, 34–43 (1986)
24. Mokhtarian F., Mackworth, A.K.: A theory of multi-scale, curvature-based shape representation for planar curves. IEEE Trans. Pattern Analysis and Machine Intelligence, **14**, 789–805 (1992)
25. Munsell, A. H.: A Color Notation. G. H. Ellis Co., Boston (1905)
26. Munsell, A. H.: A Pigment Color System and Notation. The American Journal of Psychology, **23**, 236-244 (1912)
27. Yerkes, R. M.: Introduction to Psychology. H. Holt, 1911
28. Peng, J., Yang, W., Cao, Z.: A Symbolic Representation for Shape Retrieval in Curvature Scale Space. Proc. of IEEE Int. Conf. on Computational Intelligence for Modelling Control and Automation and Int. Conf. on Intelligent Agents, Web Technologies and Internet Commerce, (2006)
29. Rosten, E., Drummond, T.:Machine learning for high-speed corner detection. Proc. of European Conference on Computer Vision, pp. 430–443 (2006)
30. Smith, A. R.: Color Gamut Transform Pairs. Computer Graphics **12**(3) August (1978)
31. Sobel, I., Feldman, G.: A 3x3 Isotropic Gradient Operator for Image Processing. Presented at a talk at the Stanford Artificial Project in 1968, unpublished but often cited, orig. in Pattern Classification and Scene Analysis, R. Duda and P. Hart, John Wiley and Sons, pp. 271–2, (1973)
32. Smith, S.M., Brady, J.M.: SUSAN – a new approach to low level image processing. Int. J. of Computer Vision, **23**, 45–78 (1997)
33. Techblog: http://www.techeblog.com/index.php/tech-gadget/samsungs-200000-machine-gun-sentry-robot
34. Tomasi C., Manduchi, R.: Bilateral Filtering for Gray and Color Images. Proc. IEEE Int. Conf. on Computer Vision (1998)
35. Trajkovic, M., Hedley, M.: Fast corner detection. Image and Vision Computing, **16**, 75–87 (1998)
36. Zhang, Z.Y.: Iterative point matching for registration of free-form curves and surfaces. Int. J. on Computer Vision, **13**, 119–152 (1994)

37. Zhong, B., Liao, W.: A Hybrid Method for Fast Computing the Curvature Scale Space Image. Proc. of the Geometric Modeling and Processing, (2004)

http://www.springer.com/978-0-387-88776-0

Multimedia Techniques for Device and Ambient Intelligence

Damiani, E.; Jeong, J. (Eds.)
2009, XIV, 201 p. 119 illus., 30 illus. in color., Hardcover
ISBN: 978-0-387-88776-0