

Chapter 2

Low-level Image Processing for Action Representations

2.1 Low-level Image Processing for Action Representations

In this book, readers are assumed to have basic knowledge of various low-level image processing. Therefore, this chapter will not cover everything that is related to action representations and understanding. However, we cite some issues, which we consider more relevant and important for action analysis.

2.2 Pre-processing Steps

Haar-like features:

Haar-like features are well-known image features, which are used in real-time face detector. The name of these rectangular Haar-like features [28] arises from their intuitive similarity with Haar wavelets. Later, others introduce the concept of tilted (45 degree) Haar-like features and generic rotated Haar-like features. Various wavelets are also exploited for image analysis.

Image pyramids:

Image pyramids are used for multi-resolution and multi-scale representation, especially in computing optical flow. By splitting an image into various levels of lower resolution and doing analysis in lower levels make it faster. Geometric transformations of an image is another important area.

Structure from motion:

Structure from motion (SFM) is the process of recovering or finding 3D structure from 2D images and it is a very important research area in computer vision. This chapter presents more on the SFM and key approaches.

Filtering:

Different filtering approaches are available for image processing, e.g., in spatial domain — low-pass filters for smoothing/ blurring (demonstrates smooth areas by removing *fine* details), high-pass filters for sharpening (demonstrate edges, noises, details — by highlighting *fine* details), averaging filter, median filters, max filter, min filter, box filter, etc.; and in frequency domain — Butterworth low-pass filter, Gaussian low-pass filter, high-pass filter, Laplacian in the frequency domain, etc. Edge detections are mentioned above. In many cases, initially, images are smoothed by employing Gaussian or other low-pass filtering schemes.

Median filtering:

Median filter is a well-known and widely used filtering scheme. We can exploit the non-linear median filter to filter out noise. Median filtering reduces noise without blurring edges and other sharp details. Median filtering is particularly effective when the noise pattern consists of strong, spike-like components (e.g., *salt-and-pepper* noise). The median filter considers each pixel in the image and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighboring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all pixel values from the surrounding neighborhood into numerical order. Then replace the pixel being considered by the middle/median pixel of that block. An image is passed into a median filter to smooth noisy patterns and hence we can achieve a smoothed image.

2.3 Segmentation and Extraction

This section concentrates on various feature detection and extraction approaches, which are widely employed in various methods for action representation for understanding and recognition.

2.3.1 Feature Detection from an Image

What is a *feature*?

What constitutes a *feature*?

It can not be clearly defined and hence, what constitutes a feature varies depending on the application. However, edges, corners, interest points, blobs, regions of interest, etc. are typically considered as image features and therefore, in image processing context, we try to extract one or more of these image features and analyze them for further processing. Note that in the presence of occlusion, shadows and image-noise, features may not find proper correspondence to the edge locations and the corresponding features.

2.3.2 Edge Detection

An edge can be defined as the points — where there is a sharp change in pixel values or gradient. Edge detection is important in many applications, and there are some edge-based representations for action analysis. Edge detection can be achieved in an image by various approaches employing,

- Gradient operators,
- Canny edge detectors,
- Sobel operators,
- Prewitt operators,
- Smallest Univalued Segment Assimilating Nucleus (SUSAN),
- Harris and Stephens / Plessey operators,
- Roberts operators,
- Laplacian operators,
- Krish operators,
- Isotropic edge operators, etc.

2.3.3 Corner Points

Corner points in an image are sometimes referred to as *interest points*. These corner points or interest points are very important cues for recent various methods related to action representation and image registration. Based on interest points, several smart local spatio-temporal interest point-based approaches are proposed with significantly good performances. These are covered in later chapters. Among various corner feature detectors,

- Features from Accelerated Segment Test (FAST) [3],
- Laplacian of Gaussian (LoG) [38, 419],
- Difference of Gaussians (DoG — DoG is an approximation of LoG) [559],
- Smallest Univalued Segment Assimilating Nucleus (SUSAN) [41],
- Trajkovic and Hedley corner detector (similar approach to SUSAN) [40],
- Accelerated Segment Test (AST)-based feature detectors,
- Harris and Stephens [39] / Plessey,
- Shi and Tomasi [145],
- Wang and Brady corner detector [37],
- Level curve curvature,
- Determinant of Hessian [419],

— etc. and some of their variants are mostly exploited in different applications.

Code:

FAST in <http://mi.eng.cam.ac.uk/~er258/work/fast.html>

2.3.4 *Blob Detectors*

Blob detectors are sometimes interrelated with corner detectors in some literatures and used the terms interchangeably. *Blob* or *regions* of interest cover the detection of those images, which are too smooth to be traced by a corner detectors. Instead of having point-like detection, a blob detector detects a region as a blob of circle or ellipse. Some important blob detectors are,

- Laplacian of Gaussian (LoG),
- Difference of Gaussians (DoG) [559],
- Determinant of Hessian,
- Maximally Stable Extremal Regions (MSER) [26],
- Principal Curvature-based Region Detector (PCBR) [36],
- Harris-affine [336],
- Hessian-affine [336],
- Edge-based regions (EBR) [336],
- Intensity Extrema-based Region (IBR) [336],
- Salient regions [336],
- Grey-level blobs.

2.3.5 Feature Descriptors

Various feature descriptors are proposed in order to detect and describe local features in images. Among various approaches, few of them achieved enormous attention in the community and are very widely used, e.g.,

- Scale-Invariant Feature Transform (SIFT) [35, 559],
- Speeded Up Robust Features (SURF) [341],
- Histogram of Oriented Gradients (HOG) [224],
- Local Energy-based Shape Histogram (LESH) [32],
- PCA-SIFT [340],
- Gradient Location-Orientation Histogram (GLOH) [367]

Speeded-Up Robust Features:

The SURF (Speeded-Up Robust Features) is a *scale*- and *rotation*-invariant interest point detector and descriptor. It has been employed for pedestrian navigation [418]. It approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet it can be computed and compared much faster [417]. This is achieved by relying on integral images for image convolutions, by building on the strengths of the existing leading detectors and descriptors (by using a Hessian matrix-based measure [419] for the detector, and a distribution-based descriptor), but uses a very basic approximation, just like DoG [559], which is a very basic Laplacian-based detector. In this case, instead of using a different measure for selecting the location and the scale, the determinant of the Hessian is employed for both cases. Here, the Hessian matrix is roughly approximated by using a set of box-type filters. Given a point $\mathbf{x} = (x, y)$ in an image I , the Hessian matrix $H(\mathbf{x}, \sigma)$ at position \mathbf{x} with scale σ is defined as follows,

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \quad (2.1)$$

where, $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative,

$$\frac{d^2}{dx^2}(G(\mathbf{x}; \sigma)). \quad (2.2)$$

— with the image I at point \mathbf{x} and similarly for $L_{xy}(\mathbf{x}, \sigma)$ and $L_{yy}(\mathbf{x}, \sigma)$. These Gaussian derivatives are approximated (lets denote these by $G_{xx}(\mathbf{x}, \sigma)$, $G_{xy}(\mathbf{x}, \sigma)$ and $G_{yy}(\mathbf{x}, \sigma)$) by considering box-type filters. Box filters can sufficiently approximate the Gaussian

derivatives. The weights applied to the rectangular regions are kept simple for computational efficiency. Finally we get,

$$\det(H_{\text{approximate}}) = G_{xx}G_{yy} - (\omega G_{xy})^2. \quad (2.3)$$

Here, ω is the relative weight of the filter responses and it is used to balance the expression for the Hessian's determinant. The approximated determinant of the Hessian represents the blob response in the image at location \mathbf{x} . A few modifications are proposed by Ehsan and McDonald-Maier [420] on reduction of the integral image length, and by Schweiger *et al.* [421].

Even though the HOG is similar to the Edge Orientation Histograms (EOH), shape contexts [27] and the SIFT descriptors, it differs in that it is computed on a dense grid of uniformly spaced cells and to enhance accuracy, it exploits overlapping local contrast normalization. The EOH contains various difference features, e.g., dominant orientation features and symmetry features; and these features have become popular recently for a wide variety of applications.

The RIFT [34] is a rotation-invariant generalization of SIFT. The PCA-SIFT [340] and Gradient Location-Orientation Histogram (GLOH) [367] are proposed at the top of the SIFT. Generalized Robust Invariant Feature (G-RIF) [33] is another feature descriptor that encodes edge orientation, edge density and hue information in a unified form.

2.3.6 Segmentation

Segmentation of an object *or* an area of interest *or* interesting features in an image, is done in many applications. Human detection aims at segmenting regions of interest corresponding to people from the rest of an image. It is a significant issue in a human motion analysis system since the subsequent processes such as tracking and action recognition are greatly dependent on the performance and the proper segmentation of the region of interest [641]. The changes in weather, illumination variation, repetitive motion, and presence of camera motion or cluttered environment hinder the performance of motion segmentation approaches. Active contour, normalized cuts, graph cuts are some widely employed methods for region segmentation.

Various clustering methods (K-means clustering, spectral clustering [392]) are employed

to cluster various regions for better segmentation. Moving object detection or pixel-change detection is the key to find the regions of interest from a scene. Possibilities for selecting *regions of interest* (ROI) are,

- Background subtraction,
- Image/frame/temporal differencing,
- Optical flow,
- Steaklines,
- Three-frame difference,
- Skin color,
- Edges, etc.

2.3.6.1 Background Subtraction

In the simplest level, background subtraction is a method to detect moving objects by calculating the differences between the current frame and the background image for each pixel and applying threshold to detect the areas of interest or foreground of the scene. Background subtraction is very important initial step for many vision-based problems, in order to extract the moving regions from a video. A naive approach for background subtraction is shown below and in Figure 2.1.

$$\text{Background subtracted image} = \text{Input image} - \text{Background image}.$$

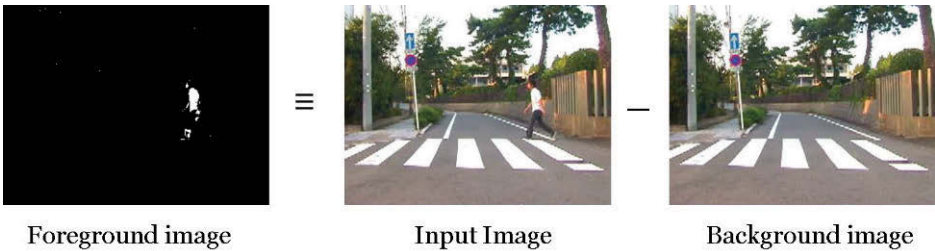


Fig. 2.1: Example of naive background subtraction method.

However, in the case of having no prior background information or image, it is a challenging task to create a background model on-the-fly. There are various methods for dynamic background modeling, e.g., by employing

- Average [13],
- Median [14],

- Gaussian Mixture Model (GMM) or Mixture of Gaussians (MoG) [12, 282],
- Parametric model,
- Non-parametric model,
- Running Gaussian average [281],
- Kernel Density Estimators (KDE) [286],
- Sequential Kernel Density Approximation (SKDA) [15],
- Mean-shift-based estimation [15],
- Combined estimation and propagation,
- Eigen-backgrounds [16], etc.

Various background subtraction techniques are reviewed by [29–31].

The background images are not fixed in moving scenes (not static background but dynamic) due to various factors, like — changes in illumination (various parts of the day have different sunlight, whereas illumination is usually static in indoor scene. Also, gradual or sudden variation of sun-light due to the presence of cloud or not); camera motion due to not having fixed camera (e.g., in ITS); movement in the background objects (e.g., movement of leaves and tree branches, sea waves, movement of objects of non-interest, etc.); changes in the background geometry (in outdoor scenes - shopping center, car-parking, road with moving cars, etc.). Therefore, based on the situation, the selection of an appropriate background subtraction model is important.

Among various methods of background subtraction, a comparative report based on some works is shown in Table 2.1.

2.3.6.2 *Frame Subtraction*

Frame to frame subtraction methods are well-known too, e.g., [287–292]. Image or, frame or, temporal differencing is defined by the differences between consecutive frames in time. Instead of subtracting a predefined or estimated background on-the-fly, a frame subtraction method considers every pair of frames of time t and $t - 1$, and extract any motion change in it — in order to find the regions of interest. Figure 2.2 shows a simple approach for frame subtraction.

Table 2.1: Background subtraction methods and their properties.

Background Subtraction Method	Speed	Memory Requirement	Accuracy*
Average [13]	Fast	High	Acceptable
Running Average [281]	Fast	Low	Acceptable
Median [14]	Fast	High	Acceptable
Mixture of Gaussians [282]	Medium	Medium	Medium
Kernel Density Estimators (KDE) [286]	Medium	High	Better
Eigen-Background [16]	Medium	Medium	Medium
Sequential Kernel Density Approximation (SKDA) [15]	Medium	Medium	Better
Standard Mean-shift [15]	Slow	High	Better
Optimized Mean-shift [15]	Medium	Medium	Better

* Regarding *accuracy* — it is difficult to analyze due to the unavailability of all methods under a benchmark and unbiased comparison of their performances. Therefore, by *acceptable*, *medium* and *better* — we rank them with lower to higher accuracy based on the existing literatures.

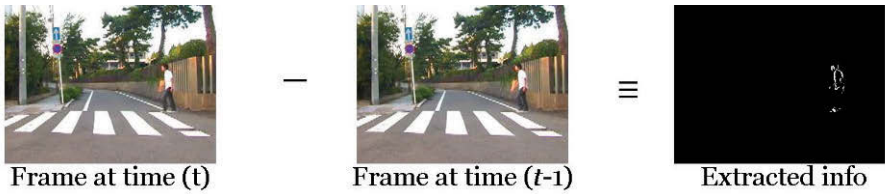


Fig. 2.2: Example of frame subtraction method.

2.3.6.3 Dense Motion Estimation

Based on Lagrangian Framework of fluid dynamics, there are 3 representations of flow [109]:

- Optical Flow or Vector Field,
- Dense Particle Trajectories or Pathlines,
- Streaklines.

Usually optical flow [293–300, 560] is used with other features, due to the fact that optical flow is noisy and it is not even consistent between frames. Therefore, optical flow offers noise and motion confusions (Figure 2.3). Even though it provides noisy patterns, many researchers have employed optical flow. Several approaches exploit the human action recognition task from optical flow (e.g., [397, 401]). In [397], for every two consecutive

frames, a local motion descriptor is calculated from the optical flow [560] orientation histograms collected inside the actor's bounding box. An action descriptor is built by weighting and aggregating the estimated histograms along the temporal axis. Few good methods for optical flow computations are — [398, 399, 560].

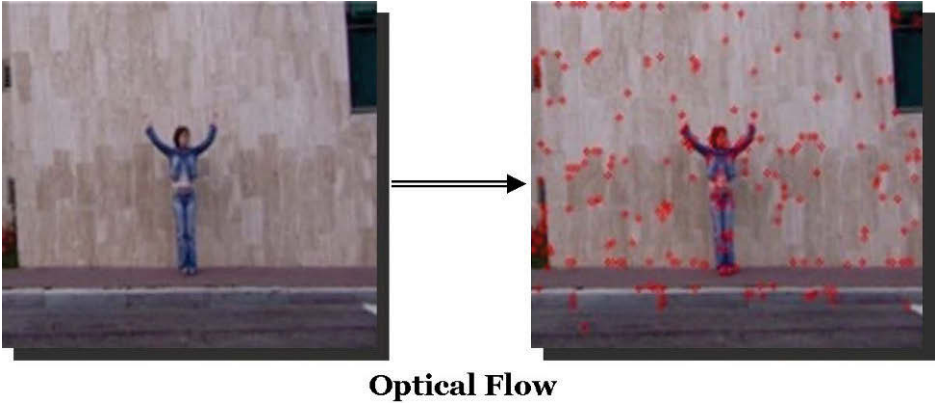


Fig. 2.3: Optical flow vectors in an action frame.

Usually, optical flow produces good results even in the presence of a small camera motion. However, it is computationally complex and very sensitive to the presence of noise and texture. For real-time processing, optical flow methods may not compute fast. Beauchemin and Barron [293] and McCane *et al.* [294] have presented various methods for optical flow. Seven different methods are tested for benchmarking optical flow methods [294]. Several real-time optical flow methods [298–300] are developed for various motion segmentation and computations. These papers can help one to choose a better optical flow method to use for a specific application.

However, in scenes with dynamic motion, where the flow changes continuously,

- Optical Flow is instantaneous, ignores the continuity of motion;
- Trajectories represent the continuity of motion, not imminent to the capture of behavior transitions.

Therefore, *streaklines* are proposed, which are instantaneous and capture the continuity of motion [108, 109]. A recent approach called streakline is proposed by Mehran *et al.* [108] based on the Lagrangian framework for fluid dynamics. The streaklines are more effective in dynamic scenes in representing the changes in motion and behavior transitions, and it can be used to solve computer vision problems involving crowd and traffic flow. As per them, streaklines are traced in a fluid flow by injecting color material, such as smoke or dye, which is transported with the flow and used for visualization [108].

In the context of computer vision, streakline may be used in a similar way to transport information about a scene, and they are obtained by repeatedly initializing a fixed grid of particles at each frame, then moving both current and past particles using optical flow. Streaklines are the locus of points that connect particles, which originate from the same initial position [108]. Streak flows encapsulate motion information of the flow for a period of time [108]. This resembles the notion of particle flow (equivalent to average optical flow), where advection of a grid of particles over a window of time provides information for segmenting the crowd motion (Figure 2.4) [108].

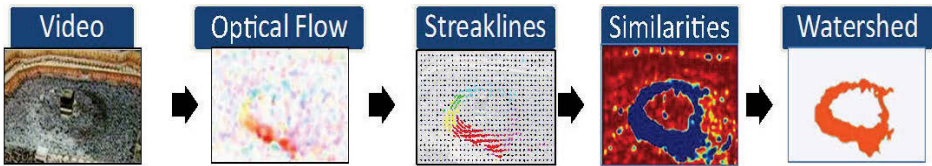


Fig. 2.4: Streaklines for crowd segmentation. (Courtesy: R. Mehran, UCF).

Streaklines are new to computer vision research. In this context, streaklines may be obtained by repeatedly initializing a grid of particles and moving all particles according to the optical flow, in the spirit of a Lagrangian fluid flow. In other words, place a particle at a point x and move the particle one time step with the flow. In the next time step, the point x is initialized with a new particle, then both particles are moved with the flow. Repeating this process on some time interval t produces particle positions from which streaklines are obtained.

Figure 2.5 gives an example of streaklines in a video. Two colors (in whitish and gray) represent the different directions of motion.

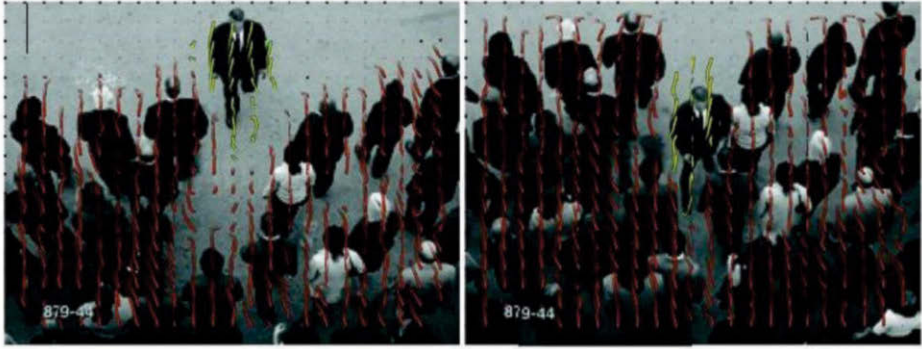


Fig. 2.5: Streaklines in a video. Two colors represent the different directions of motion. (Courtesy: R. Mehran, UCF).

Streak flow is an instantaneous vector field, which represents the accumulative motion of the scene. It resembles the temporal average of optical flow but it is more imminent. This vector field has following properties [109]:

- It averages the velocities in the direction of motion (does not average blindly) therefore, it relates to the group motion;
- Imminently reactive to changes in the scene (less lag than temporal average optical flow);
- Less noisy than optical flow (helpful in behavior recognition);

A comparison of Streak flow, optical flow, and average optical flow is given in Figure 2.6 (the Streak flow stands in the middle of average optical flow and instantaneous optical flow in capturing changes in the scene).

2.4 Local Binary Pattern

Local visual descriptors have become part of state-of-the-art systems in many areas of computer vision [366, 367]. The Local Binary Pattern (LBP) is widely exploited as an effective feature representation for various areas of face and gesture recognition and analysis. By combining the sign of the difference of central pixel intensity from those of its neighboring pixels, the Local Binary Pattern (LBP) [571, 572] implicitly encodes the micro-patterns of the input image such as flat areas, spots, lines and edges. Since the sign is invariant to monotonic photometric change, LBP is robust to lighting variation

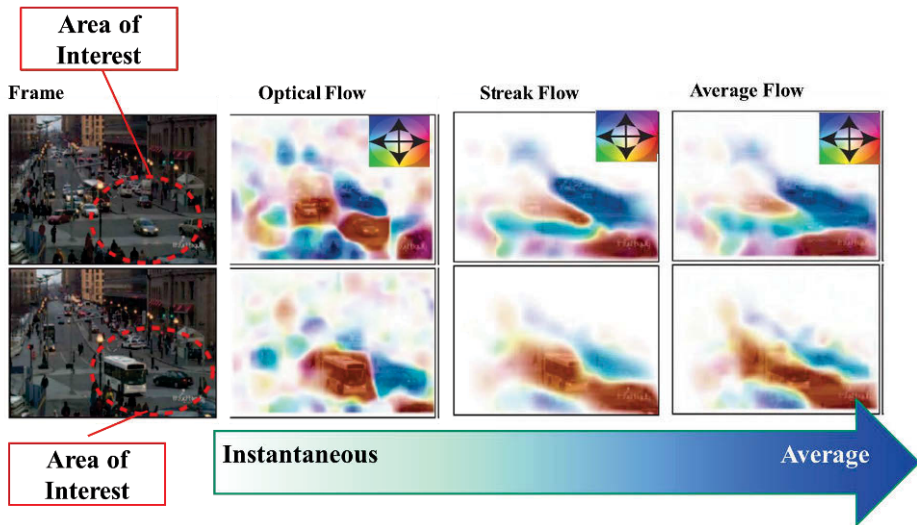


Fig. 2.6: Streaklines vs. optical flow. (Courtesy: R. Mehran, UCF).

to some extent.

Code:

LBP in <http://www.cse.oulu.fi/MVG/Downloads/LBP Matlab>

2.4.1 LBP — Variants

Several variations of the LBP have been proposed: e.g.,

- Local Ternary Patterns (LTP) [570],
- Elongated Local Binary Patterns,
- Uniform Local Ternary Pattern (ULTP) [569],
- Multi-scale LBPs [369],
- Patch-based LBPs [370],
- Center symmetric LBPs [371],
- LBPs on Gabor-filtered images [608],
- V-LGBP [372],
- LBPs on histograms of gradients [373],
- Discriminative Local Binary Patterns (DLBP) [366], etc.

Zhao and Pietikainen [606] extended LBP to the spatial-temporal domain. In order to make LBP robust to random and quantization noise in near-uniform face regions, Local Ternary Patterns (LTP) [570] have also been proposed. Tan *et al.* [570] introduce the Local Ternary Patterns (LTP) to improve the results in their Face Recognition system. It is enhanced from LBP [571] that is efficient in classifying given classes of texture with very high discrimination due to its invariant characteristics to the lighting effects. Given an image I , its LTP map is computed as follows,

$$LTP(x_c, y_c) = \sum_{t=0}^{P-1} 2^t s_{LTP}(p_t - p_c), \quad (2.4)$$

where, t is the $P - 1$ neighbors of the central pixel c , p_t and p_c are the pixel values. s_{LTP} is the indicator with three-valued codes:

$$s_{LTP}(p_t, p_c) = \begin{cases} 1 & p_t \geq p_c + r \\ 0 & |p_t - p_c| < r \\ -1 & p_t \leq p_c - r \end{cases} \quad (2.5)$$

where, pixels around the central one p_c with radius r are set to zero, the ones below this are set to -1 and the ones above this are set to $+1$. The threshold r is selected manually. For simplicity, each ternary pattern is split into positive and negative maps. These two maps are then processed as two distinguished components of descriptors containing values of either 0 or 1. Thus, their histograms and similarity metrics are computed separately. They are then combined in the final step.

Although LTP can deal with illumination variations, it is still affected when rotating the image. Luu *et al.* [569] propose an advanced feature extraction method named Uniform Local Ternary Patterns (ULTP) that is the extension of LTP. Uniform Local Ternary Patterns (ULTP) are used as the local features in [569]. Compared with LTP, ULTP has the ability of both gray-scale and rotation invariance to give more stable local feature sets [569]. Sparsely Encoded Local Descriptor (SELD) is proposed for face recognition in [605].

By combining Gabor filtering (the phase and magnitude of a multi-scale and multi-orientation Gabor wavelet are used) with LBP, Local Gabor Phase Patterns (LGPP) [608] is proposed to extend LBP to multiple resolution and orientation. Reference [321] use a framework that involves convolving the normalized face image with 40 Gabor filters that is generated by using 5 scales and 8 orientations. 40 Gabor phase maps and magnitude

maps with the same size as the image are obtained from the convolution. Uniform LBP is applied to these phase and magnitude maps by subdividing each image map spatially into blocks and histograms are computed from each block in the image map. The LBP feature vector is finally obtained by concatenating the histogram bin values [321]. This Local Gabor Binary Pattern (LGBP) is used for face representation [321].

The LBP operator in conjunction with the Gabor wavelets is used for effective feature representation [321, 323, 608]. Various methods based on *Gabor Wavelets* are:

- Elastic Bunch Graph Matching (EBGM) method [325],
- Gabor Fisher Classifier (GFC) [326],
- Local Gabor Binary Pattern Histogram Sequence (LGBPHS) [608],
- Histogram of Gabor Phase Patterns (HGPP) [607],
- Local Gabor Textons (LGT) [327],
- Learned Local Gabor Pattern (LLGP) [328],
- Local Gabor Binary Pattern Whitening PCA (LGBPWP) [329],
- Local Gabor XOR Pattern [609],
- Gabor Energy Filters,
- Local Matching Gabor method (LMG) [324, 330].

Reference [324] propose two improvements to the Local Matching Gabor (LMG) method [330]. The first one is based on weighting Gabor jets by an entropy measure between a given face and the faces in the gallery. The second improvement is in the Borda count classification stage where they propose to use a threshold to eliminate low score jets that act as noisy inputs to the classifier.

In addition, some local descriptors originally proposed for other object recognition tasks are also introduced for recognition, such as Histogram of Oriented Gradients (HOG) [611] or SIFT [559, 610].

2.5 Structure from Motion (SFM)

In this sub-section, we cover the well-known *structure from motion* (SFM) and the classical *Factorization Method* (FM) — as these are very much relevant for action understanding [185]. The reconstruction of shape and motion of a rigid object from an image sequence, usually known as the *structure from motion* (SFM) problem, is one of the

most studied problems within computer vision community. It has wide applications such as scene modeling, robot navigation, object recognition, image-based rendering, man-machine interface, virtual reality and so on. In robotics, the video camera is an increasingly popular sensor for autonomous vehicles that need to construct a 3D model of the environment for navigation and recognition purposes.

Early approaches to the SFM used a single pair of frames and have shown to be very sensitive to image noise. The key to the robustness of the SFM methods is on exploiting the rigidity of the scene across a larger set of frames. Unfortunately, although using multiple frames leads to a more constrained problem, the multi-frame formulation is also more complex — the number of unknowns grows due to the larger number of camera poses to estimate. Among the approaches to multi-frame SFM, the *factorization method* (FM) [185], introduced in the early nineties, has become popular. It is further extended by [182] to weak and para-perspective views; and is later extended in several directions, e.g., from point features [182, 185] to line features, geometric projection model, 3D shape description, recursive formulation [180] and for the perspective projection model.

Given an image sequence, suppose that there are a set of P tracked feature points over F frames. For example, a rank 4 measurement matrix \mathbf{W} [185] of dimension 30×45 , which means that 45 feature points are tracked through 15 frames. In the factorization method, an image sequence is represented as a $2F \times P$ measurement matrix \mathbf{W} . The \mathbf{W} is made up of the horizontal and vertical coordinates of the tracked feature points (P) through F frames in an image stream. This measurement matrix can be factorized into the product of two matrices \mathbf{R} and \mathbf{S} — where,

\mathbf{R} represents camera rotation matrix of size of $2F \times 3$; and

\mathbf{S} represents shape matrix in a coordinate system attached to the object centroid, of size of $3 \times P$.

When there is occlusion or failure in proper feature tracking, the \mathbf{W} is filled partially. The factorization method [185] is based on the *singular value decomposition* (SVD) technique of \mathbf{W} for scene reconstruction with an orthographic camera — where the SVD factors the measurement matrix \mathbf{W} into two matrices — \mathbf{R} and \mathbf{S} , the rotation matrix and the shape matrix respectively. The method is tested on a variety of real and synthetic images, and is shown to perform well even for distant objects, where traditional

triangulation-based approaches tend to perform poorly [182]. In real situations, feature points are visible for a limited time since they go into and out of the field of view as the camera moves (imagine a camera going around one object).

Previous approaches for computing the 3-D shape and motion from a long sequence of images usually consider,

- Whether the camera is calibrated or not, and
- Whether a projective or an affine model is used.

The factorization method has been believed to be possible under linear approximations of imaging system and without camera calibration. Lately, the original factorization method has been extended to scaled orthographic, paraperspective and perspective projection.

2.5.1 *Constraints of FM*

Both shape and motion can be factorized directly from the measurement matrix constructed from feature points trajectories under orthographic camera model. In practical applications, the measurement matrix might be contaminated by noises and contains outliers and missing values. A direct SVD to the measurement matrix with outliers would yield erroneous result.

An orthographic formulation limits the range of motions the method can accommodate [182]. The applicability of the method is therefore limited to image sequences created from certain types of camera motions. The orthographic model contains no notion of the distance from the camera to the object. As a result, shape reconstruction from image sequences containing large translations toward or away from the camera often produces deformed object shapes, as the method tries to explain the size differences in the images by creating size differences in the object. The method also supplies no estimation of translation along the camera's optical axis, which limits its applicability in certain tasks.

The factorization algorithms assume that the problem of finding correspondence between frames has been solved. But the *correspondence problem* itself is one of the most difficult fundamental problems within computer vision. In practical applications, the tracked feature points trajectories are inevitably, not always correct. Moreover, in this

approach, the nonlinear perspective projection is linearized by approximating it using orthographic projection, weak perspective projection, and paraperspective projection, which limits the range of motions that the approach can be accommodated [174]. The approximations lose some effects of perspective projection so that the accuracy of results is not always guaranteed and the range of motion is limited.

It is well-known it starts with the three largest singular values acquired by the SVD technique to factor the measurement matrix into two matrices. Then, by identifying the two matrices, shape and motion matrices can be obtained. Unfortunately, when the image noise is larger enough so that the fourth largest singular value can not be ignored, the traditional factorization method might fail to reach the accurate solution [173].

Although the factorization method is a useful technique, its applicability is, so far, limited to off-line computations for the following reasons [176]:

- (1) The method is based on a batch-type computation; that is, it recovers shape and motion after all the input images are given;
- (2) The singular value decomposition (SVD), which is the most important procedure in the method, requires $O(FP^2)$ operations for P features in F frames;
- (3) Finally, it needs to store a large measurement matrix whose size increases with the number of frames.

These drawbacks make it difficult to apply the factorization method to real-time applications. The algorithm is not practical if one intends to process data in real-time since all measurements must be available in order to compute the decomposition [176]. In this case, storage requirements grow indefinitely (the number of rows in the measurement matrix increases with the number of frames) and the estimate is produced only at the last time instant.

To recover a dense representation of the 3D shape, the factorization approach of [182, 185] may not solve the problem satisfactorily because of two drawbacks,

- Being feature-based, it would be necessary to track a huge number of features to obtain a dense description of the 3D shape. This is usually impossible because only distinguished points, as brightness corners, can be accurately tracked.

- Even if it is possible to track a large number of features, the computational cost of the SVD involved in the factorization of the measurement matrix would be very high.

2.5.2 Improvements of FM

Some advancements are proposed by Tan and Ishikawa [179, 183, 184] for recovering 3-D motions of individuals at work employing an extended measurement matrix and for separating the motions into respective motions employing k -means clustering method. This method uses at least three fixed cameras and need non-rigid points to recover though it can also recover rigid points. Another recent work [181] uses two or more moving cameras and it requires rigid and non-rigid objects for recovery of non-rigid objects. It incorporates image transfer systems efficiently.

Xi [175] presents a new algorithm for computing the SVD by linear l_1 -norm regression and applies it to structure from motion problem. It is robust to outliers and can handle missing data naturally. The linear regression problem is solved using weighted-median algorithm and is simple to implement. This robust factorization method with outliers can improve the reconstruction result remarkably. Quantitative and qualitative experiments illustrate the good performance of their approach. They review the original factorization method as defined in [174], presenting it in a slightly different manner to make its relation to the paraperspective method more apparent. They then present their paraperspective factorization method, followed by a description of a perspective refinement step. In [172], the authors propose a robust method for computing the SVD, which is based on the weighted-median algorithm. But when the rank of the input matrix is more than one, the algorithm can not give correct result. In [171], the authors propose a similar algorithm for computing subspaces, which is also based on the l_1 -norm. They optimized the l_1 -norm optimization problem using weighted-median algorithm too. But their method did not give the orthogonal left and right singular matrices explicitly. Also, since they compute the bases one by one, their method is prone to be trapped into some bad local minimum.

More recent works on the treatment of outlier trajectories include those by [169, 170] and [168]. In [170], a novel robust method for outlier detection in structure and motion recovery for affine cameras is presented. It can also be seen as an importation of the Least Median of Squares (LMedS) technique or RANSAC into the factorization frame-

work. In [168], the authors extend the above method to multiple moving objects; they fit an appropriate subspace to the detected trajectories and remove those that have large residuals. Huynh *et al.* [169] present an outlier correction scheme that iteratively updates the elements of the image measurement matrix. The magnitude and sign of the update to each element is dependent upon the residual estimated in each iteration.

Costeria and Kanade [167] propose the multi-body factorization method for reconstructing the motions and shapes of independently moving objects. They use the shape interaction matrix to separate those objects. The shape interaction matrix is also derived from the SVD of the measurement matrix. The factorization algorithm achieves its robustness and accuracy by applying the SVD to a large number of frames and feature points. The 3D motion of the camera and 3D positions of feature points are recovered by first computing the SVD of the measurement matrix. Then the metric constraints are imposed. These factorization algorithms work by linearizing the camera observation model and give good results without an initial guess for the solution. The whole procedure is linear.

Nakamura *et al.* [166] divide the image sequence into a number of blocks and the factorization method is applied to each block for obtaining a part of the 3D shape of the object. The 3D shapes reconstructed from the block of the sequence are merged into one integrated 3D Model. For the merging, position and pose of each 3D shape is adjusted based on the surface normal of the overlapped area.

Yu *et al.* [174] present a new approach based on a higher-order approximation of perspective projection based on Taylor series expansion of depth to expand the projection equations around the origin of the object coordinate system, to recover 3D shape and motion from image sequences. The accuracy of this approximation is higher than orthographic projection, weak perspective projection and paraperspective projection, so it can be used in wider circumstances in the real world in wider range of motion.

The factorization method is also extended to more general geometric projection models [182] and to more general parametric descriptions of the 3D shape [165]. These methods assume that the projections of the features are available in all frames, i.e., that they are seen during the entire video frames, which is not correct in the real-world applica-

tion as very often important regions that are seen in some frames are not seen in others, broadly due to the scene self-occlusion and the limited field of view [178]. Guerreiro *et al.* [164, 178] extend the factorization method to the more challenging scenario of observing incomplete trajectories of the feature points. In this way, they accommodate not only the features that disappear, but also features that, though not visible in the 1st image, become available later. Under this scenario, the observation matrix has missing entries. They introduce suboptimal algorithms to factor out matrices with missing data, and can compute approximation of any rank, though particularize for rank 4 matrices. It requires much smaller number of the SVD computations than the ‘filling-in’ method of [185].

Surface-based factorization [165] is simultaneously an extension and a simplification of [185]. The main ingredients of this approach are that they describe the unknown shape of the 3D rigid object by polynomial patches. Projections of these patches in the image plane move according to parametric 2D motion models. They recover the parameters describing 3D shape and 3D motion from the 2D motion parameters by factorizing a matrix that is rank 1 in a noiseless situation, not rank 3 as in [185].

The algorithm in [177] is able to maintain shape representation in a recursive way even when parts of the objects are occluded or leave the field of view. Since [177] use a recursive approach at each frame, it only has available one row of data. They show that if a point is not visible in subsequent frames, its future position can be estimated from visible points by exploiting linear dependence relations. However, this mechanism does not work backwards, that is, if a new point is selected it will not be able to reconstruct the past trajectory. The algorithm is recursive, therefore, past measurements are lost and the backward estimation process lacks data.

3D shape computation is of paramount importance in mobile robotic systems. Furthermore, real-time operation imposes particular constraints, namely recursivity. In real situations, shape computation algorithms must be able to deal with the possibility of a continuously moving camera viewpoint. This means that objects that are being viewed now will disappear in the future and new objects will come into the field of view. The ability to dynamically update 3D shape, as the scene changes without recurring to past stored measurements, is of fundamental importance. Reference [177] describe a

method that adds this capability to the well known (recursive) factorization method for shape and motion determination. This method allows the buildup of 3D image mosaics of the viewed environment. Moreover, Morita and Kanade [176] modify the method so that storing old data is no longer required to estimate motion and shape.

As stated above, the fourth largest singular value of the measurement matrix is ignored. But when noise is larger enough so that the fourth largest singular value can not be ignored, it would be difficult to get reliable results by using the traditional factorization method. In order to acquire reliable results, Hwang *et al.* [173] start with adopting an orthogonalization method to find a matrix, which is composed of three mutually orthogonal vectors. By using this matrix, another matrix can be obtained. Then, the two expected matrices, which represent shape of object and motion of camera/object, can be obtained through normalization. They concentrate on factorization method from the Rank-Theorem perspective, and improve the step of factoring by the SVD technique. According to the Rank-Theorem, it would be possible to get three mutually orthogonal vectors from a measurement matrix. Once the three mutually orthogonal vectors are identified as one matrix, it would be easy to take a form of 3D linear combination equation for obtaining another matrix. Then the identification of the two matrices can help this study to recover object shape and camera motion easily. Hwang *et al.* [173] present a form of factorization under orthographic projection, although the form of factorization also can be easily extended to other projective models. This study also provides the field of structure-from-motion with advantages, such as:

- It can robustly recover object shape and camera motion even if the emergence of image noise;
- Its computation is very fast due to a simple algorithm.

Being feature-based method [185], it would be necessary to track a huge number of features to obtain a dense description of the 3D shape and thereby computationally costly. These drawbacks motivate the extension of the factorization approach to recover a parametric description of the 3D shape [163]. Instead of tracking point-wise features, they track regions for which the motion induced on the image plane is described by a single set of parameters. This algorithm is easily extended to the scaled-orthography and the paraperspective projections by proceeding as [182] does for the original factorization method. Reference [162] further improve the algorithm and has gained two relevant advantages over the algorithms of [163, 185]. Instead of imposing a common origin for the

parametric representation of the 3D surface patches, as in [163], they allow the specification of different origins for different patches. This improves the numerical stability of the image motion estimation algorithm and the accuracy of the 3D structure recovery algorithm. Furthermore, [161] show how to compute the 3D shape and 3D motion by a simple factorization of a modified matrix that is rank 1 in a noiseless situation, instead of a rank 3 matrix as in [163, 185]. This allows the use of very fast algorithms even when using a large number of features (or regions) and large number of frames.

Computing the projective depth iteratively via iterative estimation of Euclidean shape [160] is equivalent to iterative estimation of the image coordinates by the paraperspective projection from those by the perspective projection. This method achieves accurate reconstruction of motion and shape, but is useful only in calibrated camera as Euclidean shapes are required in iterative computations. Reference [159] compute the projective depth in advance via epipolar geometry without performing iterative computation. However, the projective depth estimated via epipolar geometry is sensitive to measurement errors for feature points. Reference [158] estimate the measurement matrix containing projective depths as its elements using an evaluation function that treats all images as uniformly as possible. In this method, however, convergence of the iterative computation to estimate the measurement matrix containing projective depths takes a long time.

Recursive factorization methods can provide an updated 3D structure at each frame and at the small and fixed computational cost. Instead of one step factorization for points, a multi-step factorization method [157] is developed for lines based on the decomposition of the whole shape and motion into three separate substructures. Each of these substructures can then be linearly solved by factorizing the appropriate measurement matrices. It is also established that affine shape and motion recovery with uncalibrated affine cameras can be achieved with at least seven lines over three views.

Kurata *et al.* [180] focus on robustness against outliers, which include false matches and other objects, and the computational cost at each frame in order to apply the factorization method using point correspondences under affine projection to the real environment in real-time. Using the Least Median of Squares criterion, the method estimates the dominant 3D affine motion and can discard feature points that are regarded

as outliers. The computational cost of the overall procedure is reduced by combining this robust-statistics-based method with a recursive factorization method that can at each frame provide the updated 3D structure of an object at a fixed computational cost by using the Principal Component Analysis (PCA). Although previous work [177] also use robust statistics for recovering the epipolar geometry and the multiple projective view relation, they do not describe how to cope with the increasing computational cost as the number of frames increases.

The recursive factorization method by Fujiki *et al.* [156] compress the motion matrix, the metric constraints, and the measurement matrix by using the principal component analysis (PCA) to reduce and to fix the computational cost at each frame. To fix the world coordinates through every image, they compute the orthogonal transformation between shape matrices of two frames instead of the one between motion matrices of two frames. However, they do not evaluate the breakdown points rigorously nor consider adding new feature points. Future research will have to address these issues. For online use, they will have to improve the performance of feature tracker by speeding it up and by using the updated structure and motion.

A real-time approach is developed by [176] that develops the factorization method by regarding the feature positions as a vector time series. This method produces estimates of shape and motion at each frame. A covariance-like matrix is stored, instead of feature positions, and its size remains constant as the number of frames increases. The singular value decomposition is replaced with an updating computation of only three dominant eigenvectors. Also, the method is able to handle infinite sequences, since it does not store any increasingly large matrices, so its implementation in VLSI or DSP is feasible. Experiments using synthetic and real images illustrate that the method has nearly the same accuracy and robustness as the original method [185], except that some extra frames are required to converge. However, faster convergence in the shape space computation could be achieved using more sophisticated algorithms, such as the orthogonal iteration with Ritz acceleration instead of the basic orthogonal iteration. Also, it is possible to use scaled orthographic projection or paraperspective projection [182] to improve the accuracy of the sequential factorization method.

Factorization method based on the affine projection has a limitation in reconstruction

accuracy, and to achieve accurate reconstruction, the motion should be restricted. To overcome this problem, [155] present a recursive factorization method for the paraperspective model based on the perspective projection. This method is far superior to other ones in that it not only achieves accurate Euclidean reconstruction in a short time but also provides high stability in numerical computations. Moreover, the method produces stable reconstruction in almost all cases even if some images contain errors, because all images are treated as uniformly as possible and suitable for real-time. However, this method uses a calibrated camera.

Li and Brooks [154] propose a recursive method as a natural extension of both [185] and [176]. It estimates the shape space within a mean square errors (MSE) minimization framework. A recursive least squares (RLS) algorithm is developed for the MSE minimization, which uses as input the coordinates of feature points at each image frame. If P points are tracked through F frames, the recursive least squares method proposed by them updates the shape space with complexity $O(P)$ per frame. In contrast, [176] enables shape and motion to be updated at every frame. The cost of computing the critical shape space is $O(P^2)$ per frame. Additionally, a $P \times P$ covariance matrix is updated as part of the processing of each frame. The key SVD procedure of [185] has complexity $O(FP^2)$. Moreover, the method requires storage of a $2F \times P$ measurement matrix (its size therefore increasing with the number of frames) prior to computation of structure from motion. Hence, low computational complexity and good performance of [154] makes it suitable for real-time applications.

It is evident that the accuracy of the reconstruction will improve with better estimates of the 2D motion parameters. In turn, these estimates depend on the spatial variability of the brightness intensity pattern and on the size of the image region being tracked. The original factorization method [185] and the surface-based factorization method [165] provide equal weight to the contribution of each feature or region to the final 3D shape and 3D motion estimates. Intuitively, however, we should expect that weighting more the trajectories corresponding to ‘sharp’ features than the trajectories corresponding to features with smooth textures should lead to better overall estimates.

Aguiar and Moura [153] develop such an approach, which leads to the factorization of a modified measurement matrix rather than their original matrix in [162]. Besides

better performance, it computes the weighted estimates with no additional cost. Reference [182] consider reliability weights to address occlusion when a feature is lost, and it gives the weight to zero and uses an iterative method to recover the 3D structure. As reported in [182], the iterative method may fail to converge. Aguiar and Moura [153] show that when the weights are time-invariant, the problem can be reformulated as the non-weighted factorization of a modified matrix. Then, any method can be used to factorize this matrix. This extension is called as the weighted factorization method.

So far, in this sub-section, crucial aspects of FM for SFM are presented. More research is required for the degenerate cases including the cases where the motions are degenerate. To achieve robustness under the presence of noise and occlusion, we need to relate its level with the thresholds necessary in some of the decision making processes. Real-time applications need to be considered carefully with less computational cost in future.

2.6 Other Issues

2.6.1 *Intensity Normalization*

In this last part, we present intensity normalization of an image. There are other normalizations for size management or other purposes. In calculating the templates for various actions or activities, we get different number of frames for different actions; even the same action which is done by different subjects may have different number of frames. For example, for the Motion History Image (MHI) [434] or the Directional Motion History Image (DMHI) method [484] (details will be in later chapters), the choice of parameter τ in calculating these images is crucial. Because, if an action takes 100 frames (i.e., $\tau = 100$) by one subject, then the maximum value in the produced DMHIs will be 100 (because $\tau = 100$); whereas, if the same action is slowly accomplished by another subject (e.g., taking longer period and hence 180 frames (i.e., $\tau = 180$) for the action), then the maximum value for the developed DMHIs will be 180. This intensity variation might produce slightly different feature vectors. Therefore, incorporation of intensity normalization in computing the MHI/DMHI templates can mitigate the constraint of different number of frames to employ. Hence, we can employ a developed refined method for recognition of different actions, done by different subjects with different frame numbers with robustness. Here, the pixel intensity values of the developed templates lie in the range of $[d^{\max}, d^{\min}]$, and we wish to transform it into the range of $[0, 1]$.

Let us denote d to the original dissimilarity, and ρ to the normalized dissimilarity. Normalization can be achieved in different ways [511, 512], e.g., the following simple formulation can be used to normalize the templates into the range of $[0, 1]$. The intensity normalization transformation can be done as follows [511, 512]:

$$\rho = \frac{d - d^{\min}}{d^{\max} - d^{\min}} \quad (2.6)$$

For the MHI or DMHI-based methods, by default, $[d^{\max}] = \tau$. It changes the intensity of templates into range of $[0, 1]$. If $[d] = [d^{\min}]$, then $\rho = 0$. And if $[d] = [d^{\max}]$, then $\rho = 1$. It is evident that the relationship of the parameters of $[d] - \rho$ is linear and it depends on $[d^{\max}]$.

2.6.2 Image Matching and Correspondence Problem

Image matching has been a central research topic in computer vision over the last few decades. Correspondence or matching images having geometric and other variations is a challenging task. Various methods are proposed to solve this problem. Basically, there are two main approaches to solve the correspondence problem in images [331]:

- Model-based matching and
- Feature-based matching.

2.6.2.1 Model-based Image Matching

Model-based methods are able to solve the correspondence in difficult situations because the model encodes prior knowledge of the expected shape and appearance of an object class [331]. Active Shape Model (ASM) [342] is effective method to model the shape and appearance of objects. Several versions of Active Shape Model are,

- 3D Morphable Models (3DMM) [344],
- Active Appearance Model (AAM) [343],
- Kernel generalizations [345, 346],
- Active Conditional Model (ACM) [331], etc.

2.6.2.2 Feature-based Image Matching

Typical approaches to correspondence involve matching feature points between images. Lowe's SIFT descriptor [559], as mentioned above, is one of the state-of-the-art methods

to construct geometric invariant features to match rigid objects. SIFT and its extensions [336, 339–341] have been successfully applied to many problems.

Alternatively, the correspondence problem can be formulated into a graph matching problem considering each feature point in the image as a node in a graph [332–335, 337, 338, 348].

2.6.3 Camera Calibration

In many cases, prior camera calibration is an important part, especially in applications where we require to reconstruct a world model, when a robot system interacts with the real world, and in stereo camera system. Usually, calibrations are done by using checkerboard patterns (Tsai grid). The most commonly used camera calibration method is the DLT (direct linear transformation) method [22].

In camera calibration, we need to estimate various camera parameters (internal and external parameters) in terms of rotation R and translation t with respect to the world coordinate system — to relate the camera's image plane's coordinate and the corresponding location of the scene plane. Some internal parameters are finding the position of image center in the image, skew factor, lens distortion (pin-cushion effect), focal length, different scaling factors for row pixels and column pixels.

For an in-depth analysis, we refer to the book by [23] and a tutorial on Tsai's camera calibration method by [21].

2.7 Think Ahead!

- (1) Write a survey on segmentation methods for action recognition.
- (2) List major background subtraction approaches and using a benchmark, compare their performances.
- (3) What are the different approaches for optical flow, used in action recognition? List up their pros and cons for various applications.
- (4) List up optical flow methods that are suitable for real-time applications.
- (5) Make a judgmental research based on the existing feature point detectors — covering *which* one is the most suitable for *what* kind of images and applications.

-
- (6) Study and evaluate the variants of Local Binary Pattern and find the most suitable approach that can be suitable for action representation.

Computer Vision and Action Recognition

A Guide for Image Processing and Computer Vision

Community for Action Understanding

Ahad, M.A.R.

2011, XXI, 211 p. 43 illus., 33 illus. in color., Hardcover

ISBN: 978-94-91216-19-0

A product of Atlantis Press