CHAPTER

16 Statistical Pattern Recognition for Multi-Camera Detection, Tracking, and Trajectory Analysis

Simone Calderara, Rita Cucchiara, Roberto Vezzani
Dipartimento di Ingegneria dell'Informazione, University of Modena and Reggio Emilia, Modena, Italy

Andrea Prati
Dipartimento di Scienze e Metodi dell'Ingegneria, University of Modena and Reggio Emilia, Emilia, Italy

Abstract

This chapter will address most aspects of modern video surveillance with reference to research conducted at University of Modena and Reggio Emilia. In particular, four blocks of an almost standard surveillance framework will be analyzed: low-level foreground segmentation, single-camera person tracking, consistent labeling, and high-level behavior analysis. The foreground segmentation is performed by a background subtraction algorithm enhanced with pixel-based shadow detection; appearance-based tracking with specific occlusion detection is employed to follow moving objects in a single camera view. Thus, multi-camera consistent labeling detects correspondences among different views of the same object. Finally, a trajectory shape analysis for path classification is proposed.

Keywords: probabilistic tracking, consistent labeling, shape trajectory analysis, distributed video surveillance

16.1 INTRODUCTION

Current video surveillance systems are moving toward new functionalities to become smarter and more accurate. Specifically, path analysis and action recognition in human surveillance are two very active areas of research in the scientific community. Moreover, with the high proliferation of cameras installed in public places has come a surge of algorithms for handling distributed multi-camera (possibly multi-sensor) systems.

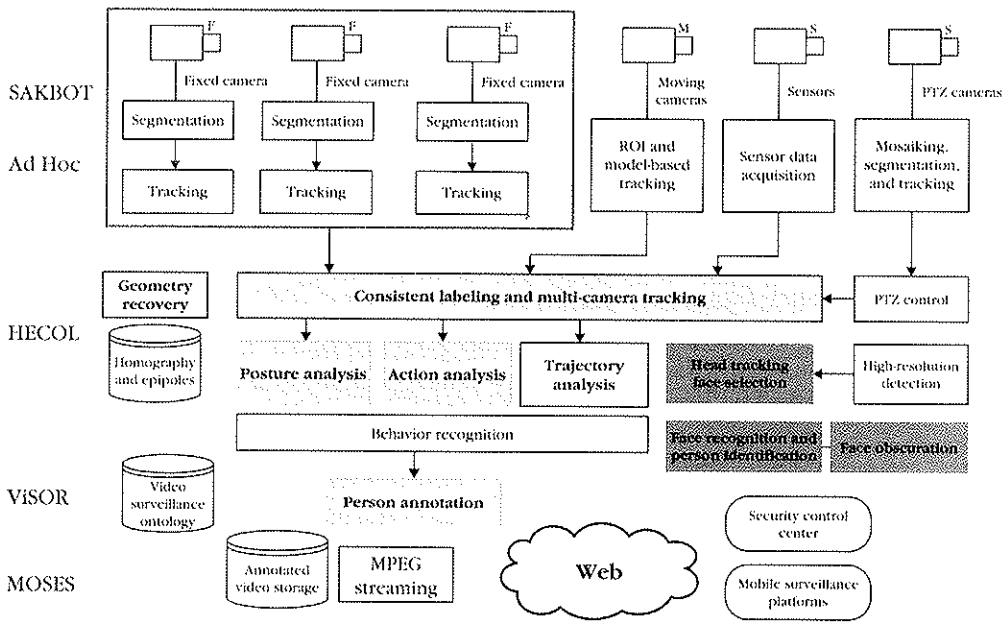


FIGURE 16.1

MPEG Imagelab video surveillance solution.

This chapter will summarize the work done at Imagelab,[1] University of Modena and Reggio Emilia, in the last ten years of research in video surveillance. Figure 16.1 shows the modular architecture we developed. The architecture exploits several libraries (collected in the Imagelab library) for video surveillance, written in C++, that provide direct interoperability with OpenCV (state-of-the-art computer vision open source tools). From the bottom up, the lowest level is the interface with sensors (the top of Figure 16.1). Traditionally video surveillance employs fixed cameras and provides motion detection by means of background suppression. Accordingly, we defined the Statistical and Knowledge-Based Object detector algorithm (SAKBOT) [1], which is very robust in many different conditions (see Section 16.2 for details). Tracking with a sophisticated Appearance-Driven Tracking Module with Occlusion Classification (Ad Hoc) [2] has been adopted in many applications of vehicle and person tracking by single cameras (see Section 16.3).

Recent advances in security call for coverage of large monitored areas, requiring multiple cameras. In cases of cameras with partially overlapped fields of view (FOVs) we propose a new statistical and geometrical approach to solve the consistent labeling problem. That is, humans (or objects) detected by a camera module should maintain their identities if acquired by other camera modules, even in cases of overlaps, partial views, multiple subjects passing the edges of the FOVs, and uncertainty. An automatic learning phase to reconstruct the homography of the plane and the epipolar lines is required to perform this task. The approach, called HECOL [3] (Homography and Epipolar-Based

[1] For further information: *http://imagelab.ing.unimore.it.*

Consistent Labeling), has been employed for real-time monitoring of public parks in Reggio Emilia.

Behavior analysis is carried out starting with the person trajectory. Learning the normal path of subjects, we can infer abnormal behaviors by means of detecting unusual trajectories that do not fit clusters of preanalyzed data. The challenge is a reliable measure of shape trajectory similarity in a large space covered by multiple cameras. In this field we recently proposed a new effective descriptor of trajectory based on circular statistics using a mixture of von Mises distributions [4].

The Imagelab architecture is enriched by a set of user-level modules. First, it includes a Web-based video and annotation repository, ViSOR [5],[2] which also provides a video surveillance ontology. Thus, MOSES (Mobile Streaming for Surveillance) is a video streaming server devoted to surveillance systems.

16.2 BACKGROUND MODELING

The first important processing step of an automatic surveillance system is the extraction of objects of interest. In particular, when cameras are installed in fixed positions this can be achieved by calculating the difference between the input frame and a model of the static content of the monitored scene—that is, the background model. Background modeling is a complex task in real-world applications; many difficulties arise from environmental and lighting conditions, micro-movement (e.g., waving trees), or illumination changes. The background model must also be constantly updated during the day because of natural intrinsic changes in the scene itself, such as clouds covering the sun, rain, and other natural artifacts.

The adopted motion detection algorithm is specifically designed to ensure robust and reliable background estimation even in complex outdoor scenarios. It is a modification of the SAKBOT system [1] that increases robustness in outdoor uncontrolled environments. The SAKBOT background model is a temporal median model with a selective knowledge-based update stage. Suitable modifications to background initialization, motion detection, and object validation have been developed. The initial background model at time t, BG_t, is initialized by subdividing the input image I into 16×16 pixel-sized blocks. For each block, a single difference over time, with input frame I_t, is performed and the number of still pixels is counted as the block's weight. The background is then selectively updated by including all blocks composed of more than 95 percent still pixels, and the initialization process halts when the whole background image BG_t is filled with "stable" blocks.

After the bootstrapping stage, the background model is updated using a selective temporal median filter. A fixed k-sized circular buffer is used to collect values of each pixel over time. In addition to the k values, the current background model $BG_t(i,j)$ is sampled and added to the buffer to account for the last reliable background information available. These $n = k + 1$ values are then ordered according to their gray-level intensity, and the median value is used as an estimate for the current background model.

[2] For further information: *http://www.openvisor.org*.

The difference between the current image I_t and the background model BG_t is computed and then binarized using two different local and pixel-varying thresholds: a low-threshold T_{low} to filter out the noisy pixels extracted because of small intensity variations; and a high-threshold T_{high} to identify the pixels where large intensity variations occur. These two thresholds are adapted to the current values in the buffer:

$$\begin{aligned} T_{\text{low}}(i,j) &= \lambda \left(b_{\frac{k+1}{2}+l} - b_{\frac{k+1}{2}-l} \right) \\ T_{\text{high}}(i,j) &= \lambda \left(b_{\frac{k+1}{2}+h} - b_{\frac{k+1}{2}-h} \right) \end{aligned} \tag{16.1}$$

where b_p is the value at position p inside the ordered circular buffer b of pixel (i,j), and λ, l, and h are fixed scalar values. We experimentally set $\lambda = 7$, $l = 2$, and $h = 4$ for a buffer of $n = 9$ values. The final binarized motion mask M_t is obtained as a composition of the two binarized motion masks computed respectively using the low and high thresholds: A pixel is marked as foreground in M_t if it is present in the low-threshold binarized mask and it is spatially connected to at least one pixel present in the high-threshold binarized mask.

Finally, the list MVO_t of moving objects at time t is extracted from M_t by grouping connected pixels. Objects are then validated, jointly using color, shape, and gradient information to remove artifacts and objects caused by small background variations, and invalid objects are directly injected into the background model (see Calderara et al. [6] for further details).

An object-level validation step is performed to remove all moving objects generated by small motion in the background (e.g., waving trees). This validation accounts for joint contributions coming from the objects' color and gradient information.

The gradient is computed with respect to both spatial and temporal coordinates of the image I_t:

$$\begin{aligned} \frac{\partial I_t(i,j)}{\partial(x,t)} &= I_{t-\Delta t}(i-1,j) - I_t(t+1,j) \\ \frac{\partial I_t(i,j)}{\partial(y,t)} &= I_{t-\Delta t}(i,j-1) - I_t(i,j+1) \end{aligned} \tag{16.2}$$

In the case of stationary points, the past image samples $I_{t-\Delta t}$ can be approximated with the background model BG_t; then the gradient module G_t is computed as the square sum of all components.

$$\begin{aligned} \frac{\partial I_t(i,j)}{\partial(x,t)} &= BG_t(t-1,j) - I_t(i+1,j) \\ \frac{\partial I_t(i,j)}{\partial(y,t)} &= BG_t(i,j-1) - I_t(i,j+1) \\ G_t &= \left\{ g(i,j) \,|\, g(i,j) = \sqrt{\left\| \frac{\partial I_t(i,j)}{\partial(x,t)} \right\|^2 + \left\| \frac{\partial I_t(i,j)}{\partial(y,t)} \right\|^2} \right\} \end{aligned} \tag{16.3}$$

This joint spatio-temporal gradient module is quite robust against small motions in the background, mainly thanks to the use of temporal partial derivatives. Moreover, the joint spatio-temporal derivative makes the gradient computation more informative, since

it also detects nonzero gradient modules even in the inner parts of the object as well as on the boundaries as performed by common techniques.

Given the list of moving objects MVO_t, the gradient G_t, for each pixel (i,j) of a moving object MVO_t, and the gradient (in the spatial domain) of the background GBG_t are compared in order to evaluate their mutual coherence. This *gradient coherence* GC_t is evaluated over a $k \times k$ neighborhood as the blockwise minimum of absolute differences between the current gradient values G_t and the background gradient values GBG_t in the considered block.

To ensure a more reliable coherence value even when the gradient module is close to zero, we combine the gradient coherence with a *color coherence* contribution CC_t, computed blockwise as the minimum of the Euclidean norm in the RGB space between the current image pixel color $I_t(i,j)$ and the background model values in the considered block centered at (i,j). The overall validation score is the normalized sum of the per-pixel validation score, obtained by multiplying the two coherence measures. Objects are validated by thresholding the overall coherence, and pixels belonging to discarded objects are labeled as part of the background.

Since shadows can negatively affect both background model accuracy and object detection, they are removed based on chromatic properties in the HSV color space [7]. The blobs classified as shadow are not tracked as the validated objects. They are not considered as background either and also they are not used for the background update.

One of the problems in selective background updating is the possible creation of ghosts. The approach used to detect and remove ghosts is similar to that used for background initialization, but at a regional rather than a pixel level. All the validated objects are used to build an image called $A_t(i,j)$ that accounts for the number of times a pixel is detected as unchanged by single difference.

A valid object MVO_t^h is classified as a ghost if

$$\frac{\sum_{(i,j)\in MVO_t^h} A_t(i,j)}{N_t^h} > T_{\text{ghost}} \tag{16.4}$$

where T_{ghost} is the threshold of the percentage of points of the MVO_t^h unchanged for a sufficient time, and N_t^h is the area in pixels of MVO_t^h.

16.3 SINGLE-CAMERA PERSON TRACKING

After being identified, moving objects should be tracked over time. To this end an appearance-based tracking algorithm is used since it is particularly suitable to video surveillance applications.

Appearance-based tracking is a well-established paradigm with which to predict, match, and ensure temporal coherence of detected deformable objects in video streams. These techniques are very often adopted as a valid alternative to approaches based on 3D reconstruction and model matching because they compute the visual appearance of the objects in the image plane only, without the need of defining camera, world, and object models. Especially in human motion analysis, the exploitation of *appearance models* or

templates is straightforward. Templates enable the knowledge of not only the location and speed of visible subjects but also their visual aspect, their silhouette, or their body shape at each frame. Appearance-driven tracking is often employed in video surveillance, particularly of humans, and in action analysis and behavior monitoring to obtain precise information about visible and nonvisible body aspects at each instant [8–10].

This section provides a formal definition of our approach, called Appearance-Driven Tracking with Occlusion Classification (Ad Hoc). The tracking problem is formulated in a probabilistic Bayesian model, taking into account both motion and appearance status. The probabilistic estimation is redefined at each frame and optimized as a MAP (maximum a posteriori) problem so that a single solution for each frame is provided in a deterministic way. We do not track each object separately; rather, the whole object set is considered in a two-step process. A first top-down step provides an estimate of the best positions of all objects, predicting their positions and optimizing them in a MAP algorithm according to pixel appearance and a specifically defined *probability of nonocclusion*. A second bottom-up step is discriminative, since each observation point is associated with the most probable object. Thus, the appearance model of each object point is selectively updated at the pixel level in the visible part, ensuring high reactivity in shape changes.

This section also describes a formal model of *nonvisible regions* that are nonnegligible parts of the appearance model unobservable in the current frame, where the pixel-to-object association is not feasible. Nonvisible regions are classified depending on the possible cause: *dynamic occlusions*, *scene occlusions*, and *apparent occlusions* (i.e., only shape variations).

16.3.1 The Tracking Algorithm

Even if Ad Hoc tracking works at a pixel level, the central element in the system is the object O, which is described by its state vector $O = \{\{o_1, \ldots, o_N\}, \vec{c}, \vec{e}, \Pi\}$, where $\{o_i\}$ is the set of N points that constitute the object O; $\vec{c}$ and $\vec{e}$ are respectively the object's position with respect to the image coordinate system and the velocity of the centroid; and Π is the probability of O being the foremost object—that is, the *probability of nonocclusion*. Each point o_i of the object is characterized by its position (x, y) with respect to the object centroid, by its color (R, G, B), and by its likelihood α of belonging to the object.

The scene at each frame t is described by a set of objects $\mathcal{O}^t = \{O_1, \ldots, O_M\}$, which we assume are generating the foreground image $F^t = \{f_1, \ldots, f_L\}$—that is, the points of MOV_t extracted by any segmentation technique. Each point f_i of the foreground is characterized by its position (x, y) with respect to the image coordinate system and by its color (R, G, B). The tracking aim is to estimate the set of objects $\mathcal{O}^{t+1}$ observed in the scene at frame $t + 1$, based on the foregrounds so far extracted. In a probabilistic framework, this is obtained by maximizing the probability $P(\mathcal{O}^{t+1}|F^{0:t+1})$, where the notation $F^{0:t+1} \doteq F^0, \ldots, F^{t+1}$. To perform this MAP estimation, we assume a first-order Markovian model, meaning that $P(\mathcal{O}^{t+1}|F^{0:t+1}) = P(\mathcal{O}^{t+1}|F^{t+1}, \mathcal{O}^t)$. Moreover, by using Bayes' theorem, it is possible to write

$$P(\mathcal{O}^{t+1}|F^{t+1}, \mathcal{O}^t) \propto P(F^{t+1}|\mathcal{O}^{t+1})P(\mathcal{O}^{t+1}|\mathcal{O}^t)P(\mathcal{O}^t) \tag{16.5}$$

Optimizing equation 16.5 in an analytic way is not possible, as this requires testing all possible object sets by changing their positions, appearances, and probabilities of nonocclusion. As this is definitely not feasible, we break the optimization process into two steps: locally optimizing the position and then updating appearance and the probability of nonocclusion.

Position Optimization

The first task of the algorithm is the optimization of the centroid position for all objects. In equation 16.5 the term $P(\mathcal{O}^t)$ may be set to 1, since we keep only the best solution from the previous frame. The term $P(\mathcal{O}^{t+1}|\mathcal{O}^t)$, the motion model, is provided by a circular search area of radius r around the estimated position $\hat{c}$ of every object. $P(\mathcal{O}^{t+1}|\mathcal{O}^t) = \frac{1}{\pi r^2}$ inside the search area and equals $\mathcal{O}$ outside.

To measure the likelihood of a foreground being generated by an object, we define a relation among the corresponding points of F and O with a function $g_O : F \rightarrow O$, and its domain $\tilde{F}_O$, which is the set of foreground points matching the object's points. We may then define $\tilde{F} = \bigcup_{O \in \mathcal{O}} \tilde{F}_O$—that is, the set of foreground points that match at least one object. In the same way we call $\tilde{O}$ the co-domain of the function g_O, which includes the points of O that have a correspondence in $\tilde{F}$. (See Figure 16.2(a).) Since the objects can be overlapped, a foreground point f can be in correspondence with more than one object O, and thus we can define the set $\mathcal{O}(f)$ as $\mathcal{O}(f) = \{O \in \mathcal{O} : f \in \tilde{F}_O\}$. The term $P(F^{t+1}|\mathcal{O}^{t+1})$ is given by the likelihood of observing the foreground image given the objects positioning, which can be written as

$$P(F^{t+1}|\mathcal{O}^{t+1}) = \prod_{f \in \tilde{F}} \left[\sum_{O \in \tilde{O}(f)} P\left(f | g_O(f)\right) \cdot \Pi_O \right] \tag{16.6}$$

obtained by adding, for each foreground pixel f, the probability of being generated by the corresponding point $o = g_O(f)$ of every matching object $O \in \mathcal{O}(f)$, multiplied by its nonocclusion probability, Π_O.

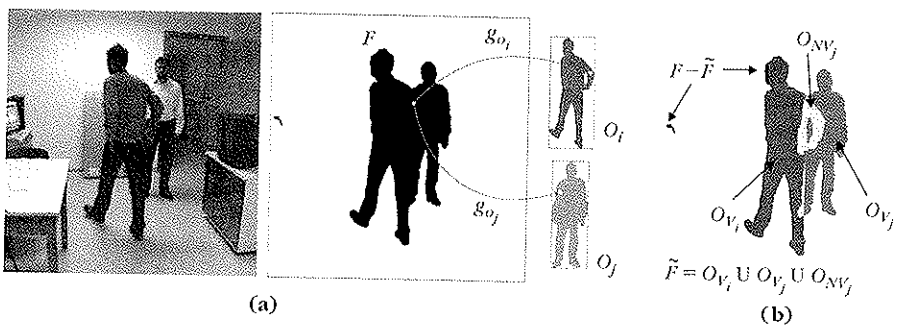


FIGURE 16.2

(a) Domain and co-domain of the function g_O, which transforms the coordinates of a foreground pixel $x \in F$ into the corresponding object coordinates. (b) Visible and nonvisible parts of an object. $\tilde{F}$ is the foreground part not covered by an object.

The conditional probability of a foreground pixel f, given an object point o, is modeled by a Gaussian distribution, centered on the RGB value of the object point:

$$P(f|o) = \frac{1}{(2\pi)^{3/2}|\Sigma|^{1/2}} e^{-1/2(\bar{f}-\bar{o})^T\Sigma^{-1}(\bar{f}-\bar{o})} \cdot \alpha(o) \tag{16.7}$$

where $\bar{(\cdot)}$ and $\alpha(\cdot)$ give the RGB color vector and the α component of the point, respectively, and $\Sigma = \sigma^2 I_3$ is the covariance matrix in which the three color channels are assumed to be uncorrelated and with fixed variance σ^2. The choice of sigma is related to the amount of noise in the camera. For our experiments we chose $\sigma = 20$.

From a computational point of view, the estimation of the best objects' alignment requires at most $(\pi r^2)^M$ evaluations. It is reasonable to assume that the contribution of the foremost objects in equation 16.6 is predominant, so we locally optimize the function by considering only the foremost object for every point. The algorithm proceeds as follows:

1. A list of objects sorted by probability of nonocclusion (assuming that this is inversely proportional to the depth ordering) is created.
2. The first object O is extracted from the list and its position $\vec{c}$ is estimated by maximizing the probability:

$$P(\tilde{F}|O) \propto \prod_{f \in \tilde{F}_O} P(f|g_O(f)) \tag{16.8}$$

3. After finding the best $\vec{c}$, the matched foreground points are removed and the foreground set F considered in the next step is updated as $F = F \backslash \tilde{F}_O$.
4. The object O is removed from the list as well, and the process continues from step 2 until the object list is empty.

The algorithm may fail for objects that are nearly totally occluded, since a few pixels can force a strong change in the object center positioning. For this reason we introduce a confidence measure for the center estimation, to account for such a situation:

$$\text{Conf}(O) = \frac{\sum_{o \in \tilde{O}} \alpha(o)}{\sum_{o \in O} \alpha(o)} \tag{16.9}$$

If during the tracking the confidence drops under a threshold (set to 0.5 in our experiments), the optimized position is not considered reliable and thus only the prediction is used.

Pixel-to-Track Assignment

This is the second phase of the optimization of equation 18.5. In this top-down approach, once all tracks have been aligned, we adapt the remaining parts of each object state. Even in this case we adopt a suboptimal optimization. The first assumption is that each foreground pixel belongs to only one object. Thus, we perform a bottom-up discriminative pixel-to-object assignment, finding the maximum of the following probability for each point $f \in \tilde{F}$:

$$P(O \rightarrow f) \propto P(f|g_O(f)) \cdot P(g_O(f)) = P(f|g_O(f)) \cdot \alpha(g_O(f)) \tag{16.10}$$

where $P(f|g_O(f))$ is the same as in equation 16.7, and we use the symbol $\rightarrow$ to indicate that the foreground pixel f is generated by the object O. Directly from the above assignment rule, we can divide the set of object points into visible O_V and nonvisible $O_{NV} = O - O_V$ points:

$$O_V = \left\{ o \in O \,|\, \exists f = g_O^{-1}(o) \wedge \arg\max_{O_i \in \mathcal{O}} \left(P(O_i \rightarrow f) \right) = O \right\} \tag{16.11}$$

In other words, the subset O_V is composed of all points of O that correspond to a foreground pixel and that have won the pixel assignment. (See Figure 16.2(b).) The alpha value of each object point is then updated using an exponential formulation:

$$\alpha(o^{t+1}) = \lambda \cdot \alpha(o^t) + (1 - \lambda) \cdot \delta(o, O_V) \tag{16.12}$$

where $\delta(\cdot, \cdot)$ is the membership function. Equation 16.12 includes two terms: one proportional to a parameter $\lambda \in [0, 1]$ that corresponds to $P(O^{t+1}|O^t)$ and reduces the alpha value at each time step; and one proportional to $1 - \lambda$ that increases the α value for the matching visible points $P(F|O)$. Similarly, we update the RGB color of each object point:

$$\bar{o}^{t+1} = \lambda \cdot \bar{o}^t + (1 - \lambda) \cdot f \cdot \delta(o, O_V) \tag{16.13}$$

The last step in updating the object state concerns the nonocclusion probability Π. We first define the probability Po^{t+1} that on object O_i occludes another object O_j:

$$Po(O_i, O_j)^{t+1} = \begin{cases} 0 & \beta_{ij} < \theta_{\text{occl}} \\ (1 - \beta_{ij}) Po_{ij}^t & a_{ij} = 0 \\ (1 - \beta_{ij}) Po_{ij}^t + \beta_{ij} e^{\frac{a_{ji}}{a_{ij}}} & a_{ij} \neq 0 \end{cases} \tag{16.14}$$

where

$$a_{ij} = \left\| O_{V,i} \underset{g}{\cap} O_{NV,j} \right\| = \left\| g_{O_i}^{-1}(O_{V,i}) \cap g_{O_j}^{-1}(O_{NV,j}) \right\|$$
$$\beta_{ij} = \frac{a_{ij} + a_{ji}}{\left\| O_i \underset{g}{\cap} O_j \right\|} \tag{16.15}$$

a_{ij} is the number of points shared between O_i and O_j and assigned to O_i; β_{ik} is the percentage of the area shared between O_i and O_j assigned to O_i or O_j, which is less than or equal to 1 since some points can be shared among more than two objects.

The value β is used as an update coefficient, allowing a faster update when the number of overlapping pixels is high. Conversely, when the number of those pixels is too low (under a threshold θ_{occl}), we reset the probability value to zero. The probability of nonocclusion for each object can be computed as

$$\Pi(O_i)^{t+1} = 1 - \max_{O_j \in \mathcal{O}} Po(O_i, O_j)^{t+1} \tag{16.16}$$

With the probabilistic framework previously described, we can "assign and track" all foreground pixels belonging to at least one object. However, the foreground image contains points $f(\in F - \tilde{F})$, with no corresponding object because of shape changes or the entrance into the scene of new objects. We assume that a blob of unmatched

foreground points is due to a shape change if it is connected (or close to) an object, and in such a situation the considered points are added to the nearest object; otherwise, a new object is created. In both cases the α value of each new point is initialized to a predefined constant value (e.g., 0.4). Obviously, we cannot distinguish in this manner a new object entering the scene occluded by or connected to a visible object. In such a situation the entire group of connected objects will be tracked as a single entity.

16.3.2 Occlusion Detection and Classification

As a result of occlusions or shape changes, some points of an object may have no correspondence with the foreground F. Unfortunately, shape changes and occlusions require two different and conflicting solutions. To keep a memory of an object's shape even during an occlusion, the object model must be slowly updated; at the same time fast updating can better handle shape changes. To this end, the adaptive update function is enriched with knowledge of occlusion regions. In particular, if a point is detected as occluded, we freeze its color and α value instead of using equations 16.12 and 16.13.

The introduction of higher-level reasoning is necessary to discriminate between occlusions and shape changes. The set of nonvisible points O_{NV} is the candidate set for occluded regions. After a labeling step over O_{NV}, a set of nonvisible regions (of connected points) is created; sparse points or too small regions are pruned, and a final set of nonvisible regions $\{NVR_j\}$ is created. Occlusions can be classified as follows:

- *Dynamic occlusions* (R_{DO}): due to overlap by another object closer to the camera; the pixels in this region were assigned to the other object.
- *Scene occlusions* (R_{SO}): due to (still) objects included in the scene and therefore in the background model, and thus not extracted by the foreground segmentation algorithm but actually positioned closer to the camera.
- *Apparent occlusions* (R_{AO}): not visible because of shape changes, silhouette motion, shadows, or self-occlusions.

The presence of an occlusion can be inferred by exploiting the confidence value of equation 16.9, decreasing it below an alerting value since in case of occlusion the object's shape changes considerably. The occluded points x of the object model ($x \in R_{DO} or x \in R_{SO}$) should not be updated because we do not want to lose its memory. Instead, if the confidence value decreases because of a sudden shape change (apparent occlusion), not updating the object state creates an error. The solution is a *selective update* according to the region classification.

The detection of the first type of occlusion is straightforward, because we always know the position of the objects and can easily detect when two or more of them overlap. R_{DO} regions comprise the points shared between object O_k and object O_i but not assigned to O_k. To distinguish between R_{SO} and R_{AO}, the position and the shape of the objects in the background can be helpful, but are not provided with our segmentation algorithm. To discriminate between R_{SO} and R_{AO}, we exploit the set of background edges. This subset of points in the background model contains all points of high color variation, among which the edges of the objects are usually detected. In the case of R_{SO} we expect to find edge points in correspondence with the boundary between this R_{SO} and the visible part

of the track. From the whole set of nonvisible points O^t_{NV} defined in equation 16.11, we only keep those with a nonnegligible value of the probability mask in order to eliminate noise due to motion. The remaining set of points is segmented into connected regions. Then, for each region, the area weighted with the probability values is calculated and too small regions are pruned. The remaining nonvisible regions (NVR_j) belonging to an object O must be discriminated as background object occlusions and apparent occlusions.

We call $B(\cdot)$ the set of border points of a region. At the same time, the edges of the background model are computed by a simple edge detector, reinforced by probability density estimation. We could exploit a more robust segmentation technique (e.g., mean shift [11]), to extract the border of objects in the background image; in our experiments, however, edge detection has given good results and requires much less computation. Given the set of edge points $E = \{e_i\}_{i=1\ldots n}$ in the background image, a probability density estimate for the background edges can be computed using a kernel $\varphi(\mathbf{x})$ and a window h:

$$p_n(\mathbf{x}|E) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^2}\varphi\left(\frac{\|\mathbf{x}-\mathbf{e_i}\|}{h}\right) \tag{16.17}$$

The probability density for nonedges $p(\mathbf{x}|\bar{\mathbf{E}})$ can be assumed to be uniform over the same region. We can then naively compute the a posteriori probability of a pixel $\mathbf{x}$ being an edge point:

$$P(E|\mathbf{x}) = \frac{P(\mathbf{x}|E)}{P(\mathbf{x}|E) + P\left(\mathbf{x}|\bar{E}\right)} \tag{16.18}$$

where we assume equal a priori probability.

We can now compute the average a posteriori probability of the set of points $o \in BO_{NV}$ to be generated by the background edges. In particular, we are interested in the subset $\tilde{B}(O_{NV}) = B(O_{NV}) \cap B(O_V)$, which is the part of the border of O_{NV} connected to the visible part O_V. The probability estimate allows a noisy match between BO_{NV} and the edge points. If this average probability is high enough, meaning that the contour of the occluded region has a good match with the edges, we can infer that another object is hiding a part of the current object, and thus label the region R_{SO}; otherwise, R_{AO}. In other words, if the visible and the nonvisible parts of an object are separated by an edge, then plausibly we are facing an occlusion between a still object in the scene and an observed moving object. Otherwise, the shape change is more reasonable because there are no more visible points.

Figure 16.3 shows a person occluded in large part by a stack of boxes that are within the background image. Two parts of his body are not segmented and two candidate occlusion regions are generated (Figure 16.3(e)). One of them is a shadow included in the object model but now gone. In Figure 16.3(g) the borders of the NVRs are shown, with pixels that match well with the edges highlighted. In an actual occlusion due to a background object, the percentage of points that match the set of bounding points is high; thus the region is classified as R_{SO}. Conversely, for the apparent occlusion (the shadow), we have no matching pixels and consequently this region is classified as R_{AO}.

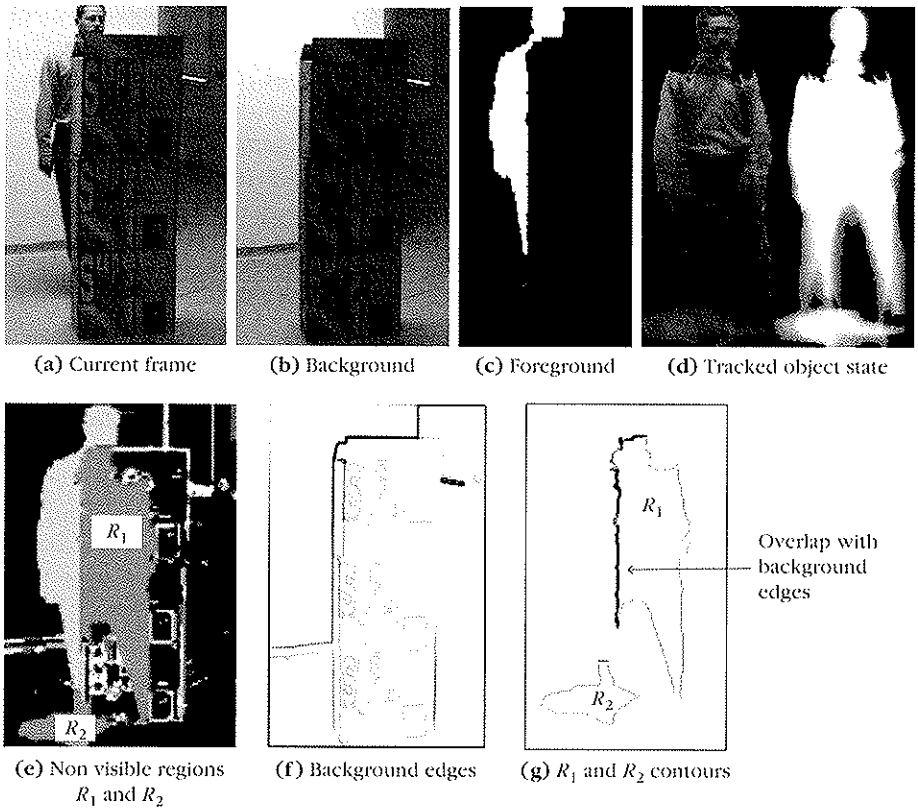


FIGURE 16.3

Example of regions classification; R_1 is classified as an R_{SO} region since there are points that match well with the edge pixels of the background. Instead, R_2 is classified as an R_{AO} region.

16.4 BAYESIAN-COMPETITIVE CONSISTENT LABELING

In large outdoor environments, multi-camera systems are required. Distributed video surveillance systems exploit multiple video streams to enhance observation. Hence, the problem of tracking is extended from a single camera to multiple cameras; thus a subject's shape and status must be consistent not only in a single view but also in space (i.e., observed by multiple views). This problem is known as *consistent labeling*, since identification labels must be consistent in time and space.

If the cameras' FOVs overlap, consistent labeling can exploit geometry-based computer vision. This can be done with precise system calibration, using 3D reconstruction to solve any ambiguity. However, this is not often feasible, particularly if the cameras are preinstalled and intrinsic and extrinsic parameters are unavailable. Thus, partial calibration or self-calibration can be adopted to extract only some of the geometrical constraints (e.g., the ground plane homography).

The consistent labeling problem, on camera networks with partially overlapping FOVs, is solved with a geometric approach that exploits the FOV relations and constraints to impose identity consistency. We call our approach HECOL (homography and epipolar-based consistent labeling). Specifically, when cameras partially overlap, the shared portion of the scene is analyzed and identities are matched geometrically. After an initial unsupervised and automatic training phase the overlapping regions among FOVs, ground plane homographies, and the epipole location for pairwise overlapping cameras are computed. The consistent labeling problem is then solved online whenever a new object τ appears in the FOV of a given camera C^1 (the superscript indicates the camera ID). The multi-camera system must check whether τ corresponds to a completely new object or to one already present in the FOV of other cameras. Moreover, it should deal with groups and identify the objects composing them. The approach described here has the advantage of coping with labeling errors and partial occlusions whenever the involved objects are present in at least one overlapped view. Using the vertical objects' inertial axis as a discriminant feature can also help to disambiguate the group detected as a single blob, exploiting the information in overlapped views.

When many objects are present in the scene and many cameras are involved, an exhaustive search may be computationally expensive. Thus, the subset of K potential matching objects satisfying the camera topology constraints is efficiently extracted by means of a graph model (called the *camera transition graph*). These K objects are combined to form the hypothesis space Γ that contains all $(2^K - 1)$ possible matching hypotheses, including both single objects and groups. A MAP estimator is used to find the most probable hypothesis $\gamma_i \in \Gamma$:

$$i = \arg\max_k \left(p(\gamma_k \mid \tau)\right) = \arg\max_k \left(p(\tau \mid \gamma_k)p(\gamma_k)\right) \tag{16.19}$$

To evaluate the maximum posterior, the prior of each hypothesis γ_k and the likelihood of the new object τ given the hypothesis must be computed. The prior of a given hypothesis γ_k is not computed by means of a specific pdf, but is heuristically evaluated by assigning a value proportional to a score σ_k. The score σ_k accounts for the distance between objects calculated after homographic warping. A hypothesis consisting of a single object then gains higher prior if the warped lower support point (i.e., the point of the object that contacts the ground plane) $\mathbf{lp}$ is far enough from the other objects' support points. On the other hand, a hypothesis consisting of two or more objects (i.e., a possible group) gains higher prior if the objects that compose it are close to each other after the warping and, at the same time, the whole group is far from other objects.

Let us suppose that a new object τ appears on camera C^1. The $\mathbf{lp}$ of each of K objects in C^2 is warped to the image plane of C^1. Likelihood is then computed by testing the fitness of each hypotheses against current evidence. The main goal is to distinguish between single hypotheses, group hypotheses, and possible segmentation errors exploiting only geometrical properties in order to avoid uncertainties due to color variation, and adopting the vertical axis of the object as an invariant feature.

The axis of the object τ can be warped correctly only with the homography matrix and knowledge of the epipolar constraints among cameras. To obtain the correct axis inclination, the vertical vanishing point (computed by a robust technique as described

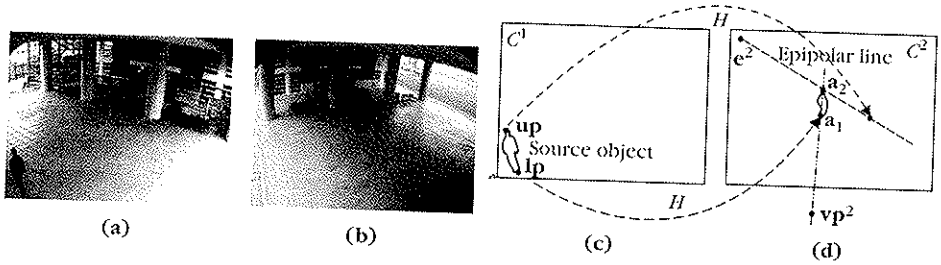


FIGURE 16.4

Example of exploiting the vanishing point and epipolar geometry to warp the axis of the object τ to the image plane of camera C^1.

in Brauer-Burchardt and Voss [12]) is then used as shown in Figure 16.4. The lower support point $\mathbf{lp}$ of τ is projected on camera C^2 using the homography matrix. The corresponding point on the image plane of camera C^2 is denoted as $\mathbf{a_1} = H\mathbf{lp}$, where H is the homography matrix from C^1 to C^2. The warped axis lies on a straight line passing through $\mathbf{vp^2}$ and $\mathbf{a_1}$ (Figure 16.4(d)). The ending point of the warped axis is computed using the upper support point up—that is, the middle point of the upper side of the object's bounding box. Since this point does not lie on the ground plane, its projection onto the image of camera C^2 does not correspond to the actual upper support point; however, the projected point lies on the epipolar line. Consequently, the axis's ending point $\mathbf{a_2}$ is obtained as the intersection between the epipolar line $\langle \mathbf{e^2}, H\mathbf{up} \rangle$ and line $\langle \mathbf{vp^2}, H\mathbf{lp} \rangle$ passing through the axis.

Based on geometrical constraints, the warped axis $\langle \mathbf{a_1}, \mathbf{a_2} \rangle$ of τ in the image plane of C^2 is unequivocally identified but its computation is not error free. To improve its robustness to computation errors, we also account for the dual process that can be performed for each of the K potential matching objects: The axis of the object in C^2 is warped on the segment $\langle \mathbf{a_1}, \mathbf{a_2} \rangle$ on camera C^1.

The measure of axis correspondence is not merely the distance between axes $\langle \mathbf{a_1}, \mathbf{a_2} \rangle$ and $\langle \mathbf{lp}, \mathbf{up} \rangle$; rather, it is defined as the number of matching pixels between the warped axis and the foreground blob of the target object—which makes it easier to define a normalized value for quantifying the matching. Accordingly, the fitness measure $\varphi_{\tau_a \to \tau_b}$ from object τ_a in generic camera C^i to object τ_b in generic camera C^j is defined as the number of pixels resulting from the intersection between the warped axis and the foreground blob of τ_b, normalized by the length (in pixels) of the warped axis itself. The reversed fitness measure $\varphi_{\tau_b \to \tau_a}$ is computed similarly by reversing the warping order. In the ideal case of correspondence between τ_a and τ_b, $\varphi_{\tau_a \to \tau_b} = \varphi_{\tau_b \to \tau_a} = 1$. However, in the case of errors in the $\mathbf{lp}$ and $\mathbf{up}$ computations, the warped axis can fall partially outside the foreground blob, lowering the fitness measure.

In the likelihood definition, we refer to *forward* contribution when fitness is calculated from the image plane in which the new object appears (camera C^1) to the image plane of the considered hypothesis (camera C^2). Thus, generalizing for hypotheses containing more than one object (group hypotheses), forward axis correspondence can be

evaluated by computing the fitness of the new object τ with all objects composing the given hypothesis γ_k for camera C^2:

$$fp_{\text{forward}}(\tau|\gamma_k) = \frac{\sum_{\tau_m \in \gamma_k^*} \varphi_{\tau \to \tau_m}}{K \cdot S_f} \tag{16.20}$$

S_f measures the maximum range of variability of the forward fitness measure of the objects inside the given hypothesis:

$$S_f = \max_{\tau_m \in \gamma_k} (\varphi_{\tau \to \tau_m}) - \min_{\tau_n \in \gamma_k} (\varphi_{\tau \to \tau_n}) \tag{16.21}$$

The use of the normalizing factor K (i.e., the number of potential matching objects on C^2) weighs each hypothesis according to the presence or absence of objects in the whole scene.

Backward contribution is computed similarly from the hypotheses space to the observed object:

$$fp_{\text{backward}}(\tau|\gamma_k) = \frac{\sum_{\tau_m \in \gamma_k} \varphi_{\tau_m \to \tau}}{K \cdot S_b} \tag{16.22}$$

where S_b is defined as

$$S_b = \max_{\tau_m \in \gamma_k} (\varphi_{\tau_m \to \tau}) - \min_{\tau_n \in \gamma_k} (\varphi_{\tau_n \to \tau}) \tag{16.23}$$

Finally, likelihood is defined as the maximum value between forward and backward contribution. The use of the maximum value ensures use of the contribution where the extraction of support points is generally more accurate and suitable for the matching.

The effectiveness of the double backward/forward contribution is evident in the full characterization of groups. The forward contribution helps solve situations when a group of objects is already inside the scene while its components appear one at a time in another camera. The backward component is useful when two people appearing in a new camera are detected as a single blob. The group disambiguation can be solved by exploiting the fact that in the other camera the two objects are detected as separate. Backward contribution is also useful for correcting *segmentation errors*, in which a person has been erroneously extracted by the object detection system as two separate objects, but a full view of the person exists from the past in an overlapped camera.

When more than two cameras overlap simultaneously it is possible to take into account more information than in the pairwise case. To account for this situation our approach is suitably modified by an additional step that selects the best assignment from all possible hypotheses coming from each camera. In detail, when a detection event occurs on C^1, for each camera C^j overlapped with C^1 the best local assignment hypothesis is chosen using the MAP framework. A second MAP estimator detects the most probable among these hypotheses. In complex scenes more hypotheses can have similar a posteriori probability but a particular view may exist where the hypothesis assignment is easier. The purpose of the second MAP stage is to choose this view, which can be easily done using the previously computed posteriors and Bayes' rule:

$$p\left(C^j \mid \tau\right) \propto p\left(\tau \mid C^j\right) = \max_{\gamma_k \in \Gamma} p(\gamma_k \mid \tau) \tag{16.24}$$

The camera posterior is evaluated for each camera C^j that overlaps with camera C^1, assuming that all overlapped camera views are equally probable. Eventually, the label is assigned to the new object according to the winning hypothesis on the winning camera. If the chosen hypothesis identifies a group, the labels of all objects composing the group are assigned as identifiers.

16.5 TRAJECTORY SHAPE ANALYSIS FOR ABNORMAL PATH DETECTION

The previous steps of our system had the main objective of keeping a person tracked in a wide area, by segmenting and tracking him in each single static camera and then exploiting consistent labeling to keep him tracked among different camera views. This global label assignment is the fundamental step for subsequent, higher-level tasks. For instance, the information provided by the multi-camera tracking system can be further analyzed to detect anomalous person paths in the scene. This task is accomplished by learning "normality" modeling path recurrence statistically as a sequence of angles modeled with a mixture of von Mises probability density functions.

In fact, after the label disambiguation process, tracking output can be exploited for further high-level reasoning on behavior. In particular, paths can be analyzed to detect anomalous events in the system. They are extracted directly from the multi-camera tracking system and homographically projected onto the ground plane. Each path is modeled as a sequence of directions computed as the angle between two consecutive points. From this approximation of the direction, a running average filter of fixed size is applied to smooth the segmentation errors and discretization effects on the direction computation.

Using a constant frame rate, we model the single trajectory T_j as a sequence of n_j directions θ, defined in $[0, 2\pi)$:

$$T_j = \{\theta_{1,j}, \theta_{2,j}, \ldots, \theta_{n_j,j}\} \tag{16.25}$$

Circular or *directional statistics* [13] is a useful framework for analysis. We adopt the von Mises distribution, a special case of the von Mises-Fisher distribution [14, 15], also known as the *circular normal* or the *circular Gaussian*. It is particularly useful for statistical inference of angular data. When the variable is univariate, the probability density function (pdf) results are

$$\mathcal{V}(\theta|\theta_0, m) = \frac{1}{2\pi I_0(m)} e^{m\cos(\theta-\theta_0)} \tag{16.26}$$

where I_0 is the modified zero-order Bessel function of the first kind, defined as

$$I_0(m) = \frac{1}{2\pi} \int_0^{2\pi} e^{m\cos\theta} d\theta \tag{16.27}$$

representing the normalization factor. The distribution is periodic so that $p(\theta + M2\pi) = p(\theta)$ for all θ and any integer M.

Von Mises distribution is thus an ideal pdf to describe a trajectory T_j. However, in the general case a trajectory is composed of more than a single main direction; having several main directions, it should be represented by a multi-modal pdf. Thus we propose the use of a mixture of von Mises (MovM) distributions:

$$p(\theta) = \sum_{k=1}^{K} \pi_k \mathcal{V}\left(\theta | \theta_{0,k}, m_k\right) \tag{16.28}$$

As is well known, the EM algorithm is a powerful tool for finding maximum likelihood estimates of the mixture parameters, given that the mixture model depends on unobserved latent variables (defining the "responsibilities" of a given sample with respect to a given component of the mixture).

The EM algorithm allows the computation of parameters for the K MovM components. A full derivation of this process can be found in Prati et al. [4].

If the trajectory T_j contains less than K main directions, some components have similar parameters. Each direction $\theta_{i,j}$ is encoded with a symbol $S_{i,j}$ using a MAP approach, that, assuming uniform priors, can be written as

$$S_{i,j} = \underset{r=1,\ldots,K}{\arg\max}\, p\left(\theta_{0,r}, m_r | \theta_{i,j}\right) = \underset{r=1,\ldots,K}{\arg\max}\, p\left(\theta_{i,j} | \theta_{0,r}, m_r\right) \tag{16.29}$$

where $\theta_{0,r}$ and m_r are the parameters of the rth components of the MovM. Each trajectory T_j in the training set is encoded with a sequence of symbols $\overline{T}_j = \{S_{1,j}, S_{2,j}, \ldots, S_{n_j,j}\}$.

To cluster or classify similar trajectories, a similarity measure $\Omega(\overline{T}_i, \overline{T}_j)$ is needed. Acquisition noise, uncertainty, and spatial/temporal shifts make exact matching between trajectories unsuitable for computing similarity. From bioinformatics we can borrow a method for comparing sequences in order to find the best inexact matching between them, accounting for gaps. Among the many techniques, we used *global alignment* [16], which is preferable to local alignment because it preserves both global and local shape characteristics. Global alignment of two sequences S and T is obtained by first inserting spaces either into or at the ends of S and T so that the length of the sequences is the same, and then placing the two resulting sequences one above the other so that every symbol or space in one of the sequences is matched to a unique symbol in the other.

Unfortunately, this algorithm is onerous in terms of computational complexity if the sequences are long. For this reason, *dynamic programming* is used to reduce computational time to $O(n_i \cdot n_j)$, where n_i and n_j are the lengths of the two sequences.

This is achieved using a tabular representation of n_j rows and n_i columns. Each element (a, b) of the table contains the alignment score of the symbol $S_{a,i}$ of sequence $\overline{T}_i$ with the symbol $S_{b,j}$ of sequence $\overline{T}_j$. This inexact matching is very useful for symbolic string recognition but it has not been used for trajectory data since it can be affected by measurement noise. Our proposal overcomes this problem because each symbol corresponds to a von Mises distribution. Thus, the score between symbols can be measured statistically as a function of the distance between the corresponding distributions. If the two distributions are sufficiently similar, the score should be high and positive; if they differ significantly, the score should be negative (a penalty). Assigning zero to the gap penalty, the best alignment can be found by searching for the alignment that maximizes the global score.

Specifically, we measured the distance between distributions p and q using the Bhattacharyya coefficient:

$$c_B(p,q) = \int_{-\infty}^{+\infty} \sqrt{p(\theta)\, q(\theta)}\, d\theta \tag{16.30}$$

It has been demonstrated [17] that if p and q are two von Mises distributions, $c_B(p, q)$ can be computed in closed form as follows:

$$\begin{aligned} c_B(S_{a,i}, S_{b,j}) &= c_B\left(\mathcal{V}\left(\theta|\theta_{0,a}, m_a\right), \mathcal{V}\left(\theta|\theta_{0,b}, m_b\right)\right) \\ &= \left(\sqrt{\frac{1}{I_0(m_a)\, I_0(m_b)}}\, I_0\left(\frac{\sqrt{m_a^2 + m_b^2 + 2m_a m_b \cos\left(\theta_{0,a} - \theta_{0,b}\right)}}{2}\right)\right) \end{aligned} \tag{16.31}$$

where it holds that $0 \leq c_B(S_{a,i}, S_{b,j}) \leq 1$.

If we assume that two distributions are sufficiently similar if the coefficient is above 0.5, and that the score for a perfect match is +2, whereas the score (penalty) for the perfect mismatch is −1 (these are the typical values used in DNA sequence alignments), then we can write the general score as follows:

$$\sigma(S_{a,i}, S_{b,j}) = \begin{cases} 2 \cdot (c_B) & \text{if } c_B \geq 0.5 \\ 2 \cdot (c_B - 0.5) & \text{if } c_B < 0.5 \\ 0 & \text{if } S_{a,i} \text{ or } S_{b,j} \text{ are gaps} \end{cases} \tag{16.32}$$

Once the score of the best global alignment is computed (as the sum of the scores in the best alignment path), it can be converted to a proper similarity measure $\Omega(\overline{T}_i, \overline{T}_j)$. This measure is used to cluster the trajectories in the training set by using the *k-medoids* algorithm [18], a suitable modification of the well-known k-means algorithm which has the benefit of computing, as a prototype of the cluster, the element that minimizes the sum of intra-class distances. In other words, at each iteration the prototype of each cluster is the member at the minimum average distance from all other members.

However, one of the limitations of *k*-medoids (as well as k-means) clustering is the choice of k. For this reason, we propose an *iterative k-medoids* algorithm. Let us set $i = 0$ and $k(0) = N_t$, where N_t is the cardinality of the training set. At initialization, each trajectory is chosen as the prototype (medoid) of the corresponding cluster. Then the following steps are performed:

1. Run the *k*-medoids algorithm with $k(i)$ clusters.
2. If there are two medoids with a similarity greater than a threshold Th, merge them and set $k(i+1) = k(i) - 1$. Increment i and go back to step 1. If all medoids have a two-by-two similarity lower than Th, stop the algorithm.

In other words, the algorithm iteratively merges similar clusters until convergence. In this way, the "optimal" number of medoids $\tilde{k}$ is obtained.

16.5.1 Trajectory Shape Classification

The described approach obtains a robust unsupervised classification of trajectories, grouped in a variable number of similarity clusters. Clusters with fewer trajectories represent the case of abnormal or (better) "infrequent" trajectory shapes. New trajectories can be classified as normal or abnormal depending on the cardinality of the most similar cluster. In this case, we cannot employ a classical *learn-then-predict* paradigm, in which the "knowledge" learned in the training phase is never updated. However, at the beginning an infrequent class of trajectories can be considered abnormal; if that class is detected often, it should be considered normal, since in our scenario the model of normality is neither a priori known nor fixed. For this reason, we employ a *learn-and-predict* paradigm in which knowledge (i.e., the trajectory clusters) is continuously updated.

Therefore, whenever a new trajectory T_{new} is collected, its statistical model is computed and compared to the cluster medoids. Based on this comparison, it can be classified as either belonging to an existing cluster or representing a new one (a class of trajectories never seen before).

To learn the trajectory model we can use the same EM algorithm described in the previous section. However, this is a very time-consuming task unsuitable for real-time trajectory classification, even though it is acceptable for offline learning. For this reason, we have derived an online EM algorithm for MovMs similar to what was proposed for a mixture of Gaussians [19].

Online EM updating is based on the concept of *sufficient statistics*. A statistic $T(\theta)$ is sufficient for the underlying parameter η if the conditional probability distribution of the data θ, given the statistic $T(\eta)$, is independent of the parameter η. Thanks to the Fisher-Neyman factorization theorem [20], the likelihood function $L_\eta(\theta)$ of θ can be factorized in two components, one independent by the parameters η and the other dependent by them only through the sufficient statistics $T(\theta)$: $L_\eta(\theta) = h(\theta)g_\eta(T(\theta))$. It was shown by Bishop [15] that in the case of distributions of the exponential family (such as Gaussian and von Mises) the factorization theorem can be written as

$$p(\theta|\eta) = h(\theta)\, g(\eta) \exp\left\{\eta^T T(\theta)\right\} \tag{16.33}$$

Considering a von Mises distribution and a set $\boldsymbol{\theta}$ of i.i.d. angles (composing the trajectory T_j), we can decompose the expression of the distribution $p(\boldsymbol{\theta}|\theta_0, m)$ as follows:

$$\begin{aligned}
\prod_{i=1}^{n_j} \frac{1}{2\pi I_0(m)} \exp\{m\cos(\theta_i - \theta_0)\} &= \frac{1}{(2\pi I_0(m))^{n_j}} \exp\left\{m \sum_{i=1}^{n_j} \cos(\theta_i - \theta_0)\right\} \\
&= \frac{1}{(2\pi I_0(m))^{n_j}} \exp\left\{m\cos\theta_0 \sum_{i=1}^{n_j} \cos\theta_i + m\sin\theta_0 \sum_{i=1}^{n_j} \sin\theta_i\right\} \\
&= \frac{1}{(2\pi I_0(m))^{n_j}} \exp\left\{\begin{bmatrix} m\cos\theta_0 \\ m\sin\theta_0 \end{bmatrix}^T \cdot \begin{bmatrix} \sum_{i=1}^{n_j} \cos\theta_i \\ \sum_{i=1}^{n_j} \sin\theta_i \end{bmatrix}\right\}
\end{aligned} \tag{16.34}$$

Thus, the sufficient statistics for a single von Mises distribution are

$$T(\boldsymbol{\theta}) = \begin{bmatrix} \sum_{i=1}^{n_j} \cos\theta_i \\ \sum_{i=1}^{n_j} \sin\theta_i \end{bmatrix}$$

In the case of a mixture of distributions belonging to the exponential family, the online updating of the mixture parameters can be obtained simply by updating the sufficient statistics (s.s.) of the mixture, computed as $T_M(\boldsymbol{\theta}) = \sum_{k=1}^{K} \gamma_k T_k(\boldsymbol{\theta})$, where $T_k(\boldsymbol{\theta})$ are the s.s. for the kth single distribution. The updating process (having observed up to the sample $(i-1)$), can be obtained as

$$T_k^i(\boldsymbol{\theta}) = \alpha(i)\gamma_k T_k(\theta_i) + (1-\alpha(i))\, T_k^{i-1}(\boldsymbol{\theta}) \tag{16.35}$$

where

$$T_k(\theta_i) = \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix}$$

In Sato [21] there is a comprehensive discussion of the value of the updating parameter $\alpha(i)$.

Once the mixture parameters have been computed, the same MAP approach described previously gives the symbol sequence $\overline{T}_{\text{new}}$. Given the set $M = \{M^1, \ldots, M^{\tilde{k}}\}$ of current medoids, $\overline{T}_{\text{new}}$ is compared with each medoid, using the similarity measure Ω, to find the most similar.

$$\tilde{j} = \underset{j=1,\ldots,\tilde{k}}{\arg\max}\ \Omega\left(\overline{T}_{M^j}, \overline{T}_{\text{new}}\right) \tag{16.36}$$

Defining the maximum similarity as $\Omega_{\max} = \Omega(\overline{T}_{M^{\tilde{j}}}, \overline{T}_{\text{new}})$, if this value is below a given threshold Th_{sim} a new cluster should be created with T_{new} and the priors (proportional to the number of trajectories assigned to the cluster) updated:

$$M^{\tilde{k}+1} = T_{\text{new}};\ p\left(M^{\tilde{k}+1}\right) = \frac{1}{N+1}$$

$$\forall i = 1, \ldots, \tilde{k} \Rightarrow p^{\text{new}}\left(M^i\right) = p^{\text{old}}\left(M^i\right)\frac{N}{N+1}$$

$$\tilde{k} = \tilde{k} + 1;\ N = N + 1$$

where N is the current number of observed trajectories.

Conversely, if the new trajectory is similar enough to one of the current medoids, it is assigned to the corresponding cluster $\tilde{j}$:

$$T_{\text{new}} \in \text{cluster}\,\tilde{j};\ p^{\text{new}}\left(M^{\tilde{k}}\right) = \frac{p^{\text{old}}\left(M^{\tilde{k}}\right)\cdot N + 1}{N+1}$$

$$\forall i = 1, \ldots, \tilde{k}, i \neq \tilde{j} \Rightarrow p^{\text{new}}\left(M^i\right) = p^{\text{old}}\left(M^i\right)\frac{N}{N+1}$$

$$N = N + 1$$

Moreover, if the average similarity of the new trajectory with regard to other medoids is smaller than the average similarity of the current medoid $M^{\tilde{j}}$, T_{new} is a better medoid than $M^{\tilde{j}}$ since it increases the separability with other clusters. Consequently, T_{new} becomes the new medoid of the cluster.

Finally, to avoid the possibility of old and rare trajectories affecting our model, we drop clusters with small priors and with no new trajectories assigned for a fixed-length time window.

16.6 EXPERIMENTAL RESULTS

Figure 16.5 shows two examples of our system in a public park (a) and on our campus (b). Detailed results and comparisons with state-of-the-art techniques can be found in the corresponding papers for background suppression [1, 6], single-camera tracking [2], and consistent labeling [3, 22]. In this chapter we focus mainly on the experimental results of trajectory shape analysis for classification and abnormality detection.

The performance evaluation was conducted on both synthetic and real data. Synthetic data is particularly useful because we can have any amount and, the ground truth is directly available; it doesn't require manual annotation. Our synthetic testing data was produced using a generator in MATLAB, which allowed us to graphically create a high number of ground truth trajectories with noise added to both single angles and their occurrences.

Figure 16.6 shows examples of the trajectory classes used in our tests. In the case of real data (see classes R1 . . . R5) the trajectories' points are extracted from the scene using the HECOL system described in Section 16.4.

Table 16.1 summarizes the performed tests. For each, the classes of trajectories used are depicted (with reference to Figure 16.6), with the asterisk representing an abnormal (infrequent) class. For testing purposes we evaluated both overall classification accuracy (ability to assign the new trajectory to the correct cluster) and normal/abnormal

FIGURE 16.5

Example multi-camera tracking results.

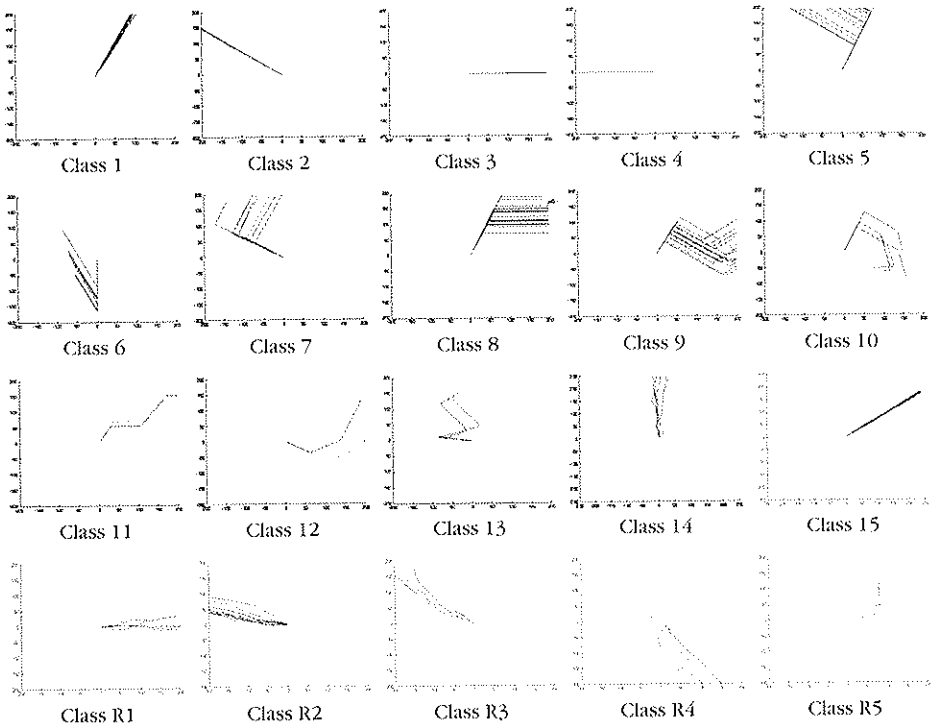


FIGURE 16.6

Example trajectory classes.

accuracy (ability to correctly classify the trajectory as normal or abnormal depending on the cardinality of the specific cluster).

Our system demonstrates optimal results both in classification and normal-abnormal detection; the results are also optimal for real data. We compared our approach with an HMM-based classification system using the similarity measure proposed in Porikli and Haga [23] (see the last two rows of Table 16.1). HMM's lower classification performance is mainly due to the overfitting problem. When little data is available, the HMM training stage fails to correctly estimate all parameters. Emission distribution and optimal number of hidden states are crucial elements that must be chosen accurately when using an HMM. This procedure can be unsupervised but it needs a large amount of data that is not always available in real scenarios. Conversely, our approach is not greatly affected by the data available because the number of parameters to estimate is significantly lower; thus it can be profitably applied in many different situations.

Acknowledgments. This chapter addressed aspects of modern video surveillance that relate to our research at the University of Modena and Reggio Emilia. This research is sponsored by the FREE SURF (Free Surveillance in a Privacy-Respectful Way) and NATO-funded BE SAFE (Behavioral Learning in Surveilled Areas with Feature Extraction) projects.

Table 16.1 Summary of the Experimental Results

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
Type of Data	Synthetic	Synthetic	Synthetic	Synthetic	Synthetic	Synthetic	Synthetic	Real
Type of Test	Periodicity	Noise	Mono-modal	Multi-modal	Sequence	Learning normality	Mixed	Mixed
Total no. Trajectory Tr	250	400	450	680	650	150	2530	520
Te	150	250	400	430	400	200	1700	430
Average Points/Trajectory	65	95	66	74	85	95	105	66
Training Set	C_3, C_4^*	C_1, C_{15}	C_1, C_2, C_3	C_1, C_2, C_5, C_6^*	C_1, C_8, C_9, C_{10}^*	C_1, C_7^*	All training	$C_{R1}, C_{R2}, C_{R3}, C_{R4}^*$
Testing Set	C_3, C_4^*	C_{14}^*, C_{15}	C_1, C_2, C_3^*, C_4^*	C_1, C_5, C_6^*, C_7^*	$C_8, C_2^*, C_9^*, C_{11}^*, C_{12}^*, C_{13}^*$	C_{13}^*, C_7	All testing	$C_{R1}, C_{R2}, C_{R4}^*, C_{R5}^*$
Classification Accuracy	100%	100%	100%	97.67%	94.59%	95%	99.60%	96.77%
Normal/Abnormal Accuracy	100%	100%	100%	100%	97.30%	100%	100%	96.77%
Classification Accuracy HMM	75%	83%	94.1%	92.67%	89.1%	94.01%	86.40%	75.19%
Normal/Abnormal Accuracy HMM	84%	94.1%	93.02%	100%	85.1%	100%	82.4%	66%

FREE SURF, funded by the Italian Ministry for University Research (MIUR), focuses on the study of innovative video surveillance solutions, bringing together systems without physical constraints (i.e., those using PTZ, freely moving cameras, and sensors) and completely respectful of privacy issues (i.e., free from legal constraints). The FREE SURF[3] research activities are performed in collaboration with the University of Firenze and the University of Palermo.

BE SAFE[4] is a NATO Science for Peace project that focuses on extracting visual features that can be used for understanding behaviors, such as potential terrorist attacks. It is carried out in collaboration with the Hebrew University of Jerusalem.

REFERENCES

[1] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, Detecting moving objects, ghosts and shadows in video streams, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (10) (2003) 1337–1342.

[2] R. Cucchiara, C. Grana, G. Tardini, R. Vezzani, Probabilistic people tracking for occlusion handling, in: Proceedings of the International Conference on Pattern Recognition, 2004.

[3] S. Calderara, A. Prati, R. Cucchiara, HECOL: Homography and epipolar-based consistent labeling for outdoor park surveillance, Computer Vision and Image Understanding 111 (1) (2008) 21–42.

[4] A. Prati, S. Calderara, R. Cucchiara, Using circular statistics for trajectory shape analysis, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2008.

[5] R. Vezzani, R. Cucchiara, ViSOR:Video surveillance on-line repository for annotation retrieval, in: Proceedings of the IEEE International Conference on Multimedia and Expo, 2008.

[6] S. Calderara, R. Melli, A. Prati, R. Cucchiara, Reliable background suppression for complex scenes, in: Proceedings of the ACM Workshop on Video Surveillance and Sensor Networks, Algorithm Competition, 2006.

[7] A. Prati, I. Mikic, M. Trivedi, R. Cucchiara, Detecting moving shadows: Algorithms and evaluation, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (7) (2003) 918–923.

[8] I. Haritaoglu, D. Harwood, L. Davis, W4: real-time surveillance of people and their activities, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 809–830.

[9] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, R. Bolle, Appearance models for occlusion handling, Image and Vision Computing 24 (11) (2006) 1233–1243.

[10] R. Cucchiara, C. Grana, A. Prati, R. Vezzani, Probabilistic posture classification for human behaviour analysis, IEEE Transactions on Systems, Man, and Cybernetics 35 (1) (2005) 42–54.

[11] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (5) (2003) 564–575.

[12] C. Brauer-Burchardt, K. Voss, Robust vanishing point determination in noisy images, in: Proceedings of the International Conference on Pattern Recognition, 2000.

[13] K. Mardia, P. Jupp, Directional Statistics, Wiley, 2000.

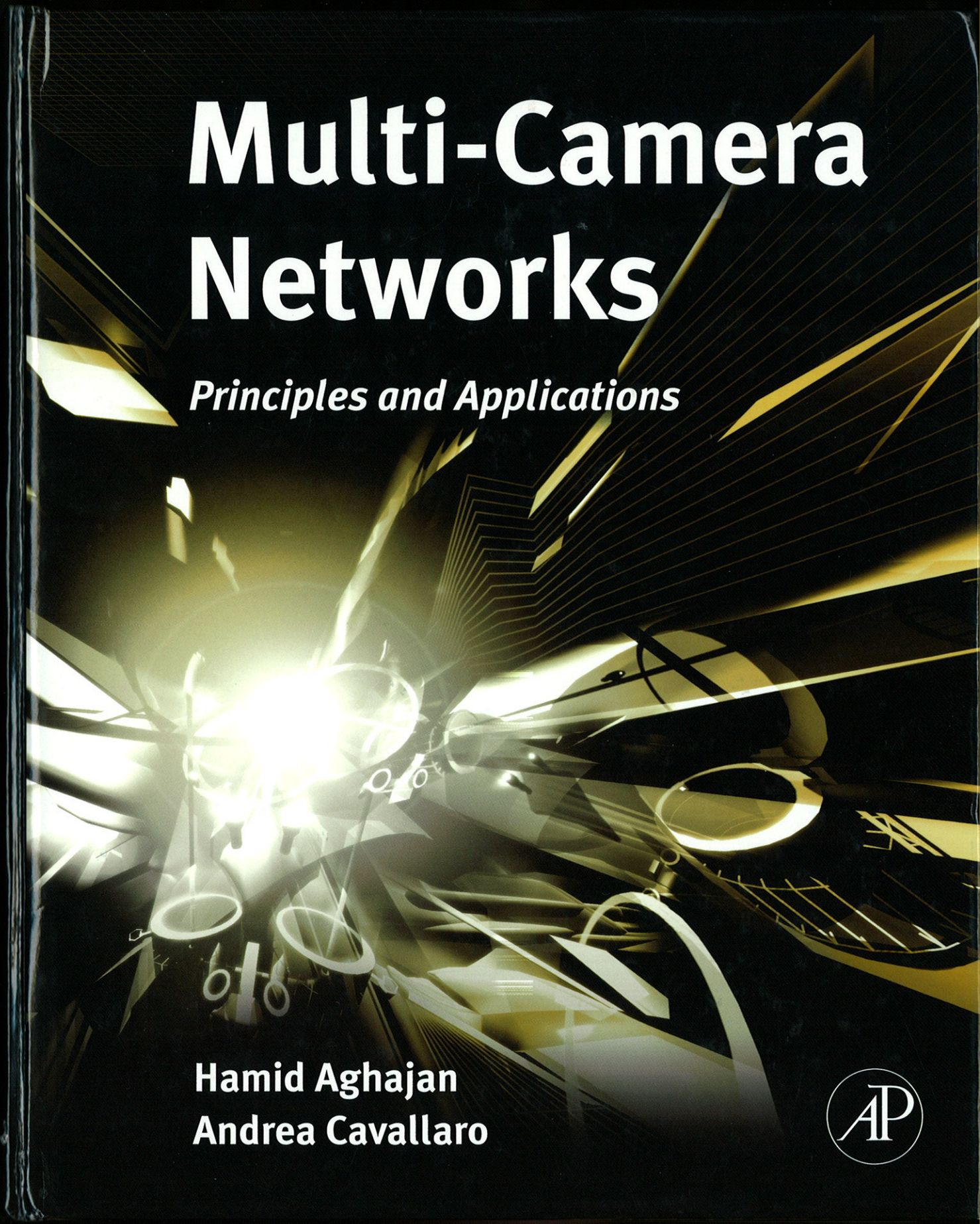
[14] R. Fisher, Dispersion on a sphere, Proceedings of the Royal Society of London, Series A 217 (1953) 295–305.

[3] *http://imagelab.ing.unimore.it/freesurf.*
[4] *http://imagelab.ing.unimore.it/besafe.*

[15] C. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag, 2006.
[16] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.
[17] S. Calderara, R. Cucchiara, A. Prati, Detection of abnormal behaviors using a mixture of von Mises distributions, in: Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance, 2007.
[18] A. Reynolds, G. Richards, V. Rayward-Smith, The Application of K-Medoids and PAM to the Clustering of Rules, Springer-Verlag, 2004.
[19] C. Stauffer, W. Grimson, Learning patterns of activity using real-time tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 747–757.
[20] G. Casella, R. Berger, Statistical Inference, 2nd edition, Duxbury Press, 2002.
[21] M. Sato, Fast learning of on-line EM algorithm. Technical Report TR-H-281, ATR Human Information Processing Research Laboratories, 1999.
[22] S. Calderara, R. Cucchiara, A. Prati, Bayesian-competitive consistent labeling for people surveillance, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2) (2008) 354–360.
[23] F. Porikli, T. Haga, Event detection by eigenvector decomposition using object and frame features, in: Proceedings of the Computer Vision and Pattern Recognition Workshop, 2004.

Multi-Camera Networks

Principles and Applications



Hamid Aghajan
Andrea Cavallaro