Chapter 6

A DISTRIBUTED MULTIPLE CAMERA SURVEILLANCE SYSTEM

T.Ellis, J.Black, M.Xu and D.Makris
Digital Imaging Research Centre (DIRC), Kingston University, UK
{t.ellis,j.black,m.xu,d.makris}@kingston.ac.uk

1. Introduction

An important capability of an ambient intelligent environment is the capacity to detect, locate and identify objects of interest. In many cases interesting can move, and in order to provide meaningful interaction, capturing and tracking the motion creates a perceptively-enabled interface, capable of understanding and reacting to a wide range of actions and activities. CCTV systems fulfill an increasingly important role in the modern world, providing live video access to remote environments. Whilst the role of CCTV has been primarily focused on rather specific surveillance and monitoring tasks (i.e. security and traffic monitoring), the potential uses cover a much wider range.

The proliferation of video security surveillance systems over the past 5-10 years, in both public and commercial environments, is extensively used to remotely monitor activity in sensitive locations and publicly accessible spaces. In town and city centres, surveillance has been acknowledged to result in significant reductions in crime. However, in order to provide comprehensive and large area coverage of anything but the simplest environments, a large number of cameras must be employed.

In complex and cluttered environments with even moderate numbers of moving objects (e.g. 10-20) the problem of tracking individual objects is significantly complicated by occlusions in the scene, where an object may be partially occluded or totally disappear from camera view for both short or extended periods of time. Static occlusion results from objects moving behind (with respect to the camera) fixed elements in the scene (e.g. walls, bushes), whilst dynamic occlusion occurs as a result of moving objects in the scene occluding each other, where targets may merge or separate (e.g. a group of people walking together).

Information can be combined from multiple viewpoints to improve reliability, particularly taking advantage of the additional information where it minimises occlusion within the field-of-view (FOV). We treat the non-visible regions between camera views as simply another type of occlusion, and employ

spatio-temporal reasoning to match targets moving between cameras that are spatially adjacent. The "boundaries" of the system represent locations from which previously unseen targets can enter the network.

To aid robust tracking across the camera network requires the system to maintain a record of each target entering the system and throughout its duration. When a target disappears from any camera FOV, motion prediction, colour identification, and learnt route patterns are used to re-establish tracking when the target reappears. Each target is maintained as a persistent object in the active database and spatial and temporal reasoning are used to detect these activities and ensure that entries are not retained for indefinite periods.

This chapter describes a multi-camera surveillance network that can detect and track objects (principally pedestrians and vehicles) moving through an outdoor environment. The remainder of this chapter is divided into four sections. The first describes the architecture of our multi-camera surveillance system. The second considers the image analysis methods for detecting and tracking objects within a single camera view. The next section deals with the integration of information from multiple cameras. The final section describes the structure of the database.

2. System architecture

The multi view tracking framework of the surveillance system has been implemented using the architecture shown in Fig. 6.1. The system comprises a set of intelligent camera units (ICU) that detect and track moving objects in 2D image coordinates. It is assumed that the viewpoint of each ICU is fixed and has been calibrated using a set of known 3D landmark points in order to establish a common world coordinate system across the camera network. Each ICU communicates with a central multi view tracking server (MVT), which integrates the information to generate global 3D track data. Individual objects and associated tracking details are stored in a central database. The surveillance database also enables offline learning and subsequent data analysis (see Chapter 7). In addition, given the query and retrieval properties of the surveillance database it is possible to generate pseudo synthetic video sequences that can be used for performance evaluation of object tracking algorithms. The surveillance system employs a centralised control strategy as shown in Fig. 6.1. The multi view tracking server creates separate receiver threads (RT) to process the data transmitted by each intelligent camera unit (ICU) connected to the surveillance network. Each ICU transmits tracking data to each RT in the form of symbolic packets through TCP/IP sockets. The database fulfills a dual role of storing data (e.g. raw track data, ground-plane trajectories, compressed video) and serving online system parameters (e.g. camera calibration data, network ID's) to each of the ICU's, supporting dynamic update of parameters during operation.

3. Motion detection and single-view tracking

The first step in the motion detection process identifies potential regions (cues) associated with object motion by using frame differencing, subtracting

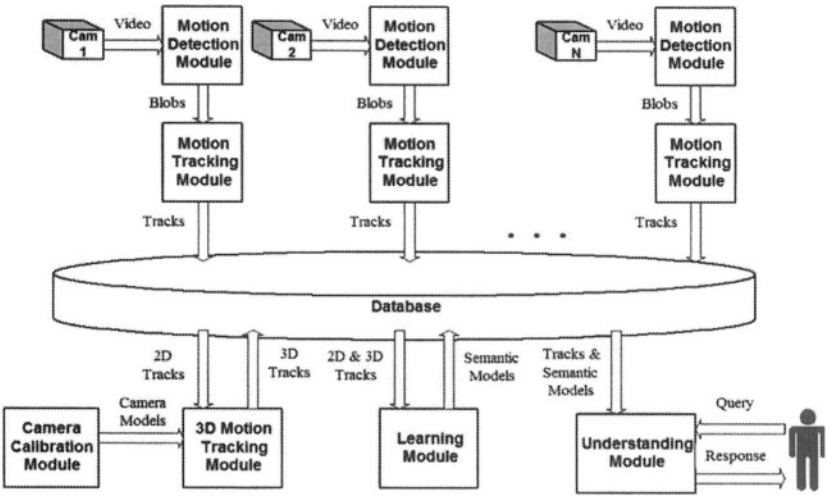

Figure 6.1. Architecture of the Kingston University Experimental Surveillance (KUES) System.

the current image from a reference or background image. This is highly efficient yet sensitive method for detecting changes in the images from fixed cameras, but relies on an accurate estimate of the background which itself may be changing due to variations in illumination (e.g. the diurnal cycle, clouds, artificial lighting), the vagaries of the weather (rain, snow etc), misleading optical effects (e.g. shadows, reflections, specularity) and a miscellaneous variety of irrelevant motion (e.g. wind-blown vegetation and flags, pictures on TV monitors).

A chromaticity-plus-intensity based Gaussian mixture model is used to model the background value for each pixel. The region-based representations of each object is tracked and predicted using a Kalman filter. To increase the robustness and accuracy of object tracking during grouping and occlusion, partial observations are used whenever available. A scene model and a Bayesian network are jointly used to interpret the occluded and exiting objects.

3.1 Motion Detection

Our system uses frame differencing for change detection, comparing each incoming frame with an adaptive background image and classifies those pixels of significant variation into foreground. The probability of observing values $\mathbf{I}$ at a pixel is modelled by a mixture of Gaussians [13]:

$$P(\mathbf{I}_k) = \sum_i \omega_k^{(i)}(\mathbf{I}_k, \mathbf{u}_k^{(i)}, \sigma_k^{(i)}) \tag{6.1}$$

where $\mathbf{u}_k^{(i)}$ is the temporal mean of the i-th distribution, $(\sigma_k^{(i)})^2$ is the trace of the covariance matrix, and $\omega_k^{(i)}$ is the weight reflecting the prior probability that the i-th distribution accounts for the data. At time k, every new pixel value is checked against the Gaussian distributions in a mixture model. For a matched distribution, the pixel measurement is incorporated in the estimate of that distribution and the weight is increased:

$$\mathbf{u}_k = (1-\rho)\mathbf{u}_{k-1} + \rho\mathbf{I}_k \tag{6.2}$$

$$\sigma_k^2 = (1-\rho)\sigma_{k-1}^2 + \rho||\mathbf{I}_k - \mathbf{u}_k||^2 \tag{6.3}$$

For unmatched distributions, their estimates remain the same but the weights are decreased. If none of the existing distributions matches the current pixel value, either a new distribution is created, or the least probable distribution for the background is replaced. The distribution(s), i_B , with the greatest weight is (are) identified as the *apriori* background model for the next frame. At time k, the set of foreground pixels identified is:

$$F_k = \{(r,c) : ||\mathbf{I}_k(r,c) - \mathbf{u}_{k-1}^{(i_B)}(r,c)|| > 2.5\sigma_{k-1}^{(i_B)}(r,c)\} \tag{6.4}$$

A major challenge for target detection in an outdoor surveillance environment comes from the flood of sunlight, to which many existing applications using intensity or *(R, G, B)* pixel representation are sensitive. Chromaticity-based representation for pixel values is a partial remedy. However, the relevant detection is noisy in poorly lit regions and may be undersegmented when part of a target has similar chromaticity to the background. Therefore, the intensity, $I = R + G + B$, and the chromaticity (r, g, b) are combined by using two separate mixtures of Gaussians to model the pixel observation [16].

Suppose $F_k^{(c)}$ and $F_k^{(i)}$ are the sets of the foreground pixels identified using chromaticity and intensity inputs, respectively. The set of final foreground pixels can be computed as the intersection between chromaticity-based foreground (dilated) and intensity-based foreground:

$$F_k = (F_k^{(c)} \oplus B) \cap F_k^{(i)} \tag{6.5}$$

where $\oplus$ denotes the morphological dilation with rectangular structuring element B. The fusion between two types of mixture models overcomes the disadvantages of using each model separately. In regions with lighting variation, few spurious chromaticity-based foreground regions are produced, which masks spurious intensity-based foregrounds due to the sensitivity to illumination changes; in poorly-lit regions, few spurious intensity-based foreground regions are detected, which masks the spurious chromaticity-based foregrounds caused by the sensitivity to noise; undersegmented chromaticity-based foregrounds, due to their similar chromaticity to the background, can be bridged or resized with the morphological dilation by B. The foreground pixels are filtered using a morphological closing (dilation-plus-erosion) operation and then clustered into regions using connected component analysis. Fig. 6.2 shows an example of foreground detection during a minor illumination change. The spurious

foreground at the top-left corner of the intensity-based result is caused by the illumination change and is suppressed in the chromaticity-plus-intensity result, yet the foreground objects equally well detected.

Figure 6.2. Foreground detection under an illumination change: (left) the original frame, (middle) intensity-based foregrounds, and (right) chromaticity-plus-intensity foregrounds.

In low light levels, when the colour quality of a CCTV camera may be severely compromised, the chromaticity-based detection can be disabled. For this, the average intensity of the input image and its variance are used to control the switching. For example, at sunset when the average intensity in a scene is very low, or on an overcast day when the average intensity is moderate but varies slowly, the chromaticity-based detection can be disabled and detection is wholly based on the intensity-based model, with a resultant saving in processing time [16].

The final step encodes the blobs detected by the connected component analysis, representing each with a measurement vector comprised of size, shape and appearance information: area, coordinates of the bounding box, blob centroid and colour histogram.

3.2 Scene Models

Because the camera is fixed, a scene model for static occlusions can be constructed for a specific camera position to support object tracking and reasoning through occlusion [5]. Whilst in this section we employ a manually-constructed scene model, chapter N describes an approach for automatically learning elements such models.. Three types of static scene occlusion are identified as (Fig. 6.3):

- *Border occlusion* (BO), outside the limits of the camera field-of-view.
- *Long-term occlusions* (LO) where objects may leave the scene earlier than expected, resulting in termination of a record in the object database. The long-term occlusion may exist at the border (e.g. buildings or vegetation) or in the middle of an image (e.g. at the entrance to a building). Without prior knowledge of these long-term occlusions, a target disappearing at an LO will be maintained in the object database for a certain of time and may then be mismatched with another target moving in front of the underlying occlusion.

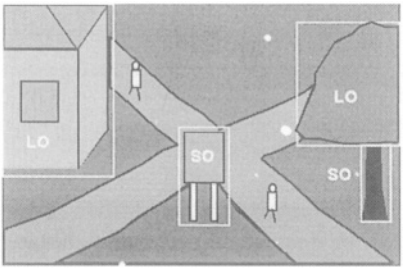

Figure 6.3. Scene model for static occlusions.

- *Short-term occlusions* (SO), where an object is temporarily occluded by a static occlusion, e.g. a tree or a road sign. Prior knowledge of these occlusions can be used to minimize tracking errors. A target moving behind a short-term occlusion is allowed to survive longer than a normal target without observation, depending on the physical dimensions of the underlying occlusion and the velocity of the target.

Each occlusion is characterised by its type (BO, LO or SO) and the region defined by its bounding box. The overlap of target predicted measurement with these static occlusions is used to decide target observability and reason about the status of targets without measurement. Currently a rectangular bounding box is used for each static occlusion to minimise the computational cost. A more accurate representation of an occlusion, e.g. using a convex polygon, is straightforward but not so important here, because the occlusion bounding boxes are only used for the prediction of termination and occlusion events. The determination of such events also depends on the result of tracking (e.g. an object fails to find a corresponding foreground region) since objects may pass in front of an LO or SO type occlusion.

3.3 Target Tracking

Each foreground region is represented by a foreground measurement vector:

$$\mathbf{f} = \begin{bmatrix} r_c & c_c & r_1 & c_1 & r_2 & c_2 \end{bmatrix}^T$$

where (r_c, c_c) is the centroid, r_1, c_1, r_2, c_2 represent the top, left, bottom and right bounding edges, respectively ($r_1 < r_2$ and $c_1 < c_2$). A foreground region may correspond to an object, a group of objects due to dynamic occlusion, or part of an object due to static occlusion. In this paper, we use $\mathbf{f}(i)$ to represent the i-th element of the vector $\mathbf{f}$, e.g. .

A Kalman filter based on a first-order motion model is used to track each object according to the object measurement vector (see section 7.1):

$$\mathbf{z} = \begin{bmatrix} r_c & c_c & r_1 & c_1 & r_2 & c_2 \end{bmatrix}^T$$

We distinguish object measurements from foreground measurements, because they are the same only for separate objects. Because our system aims to monitor pedestrians and vehicles, each target is assumed to move along a linear trajectory at constant velocity and with constant size. In practice, any minor violation of this assumption can be encoded in the process covariance matrix. The state vector used is:

$$\mathbf{x} = [\ r_c \quad c_c \quad \dot{r}_c \quad \dot{c}_c \quad \Delta r_1 \quad \Delta c_1 \quad \Delta r_2 \quad \Delta c_2\ ]^T$$

where $(\Delta r_1, \Delta r_2)$ and $(\Delta r_2, \Delta c_2)$ are the *relative* positions of the two opposite bounding box corners to the centroid. They not only incorporate height and width information, but also accurately represent the bounding box even when the centroid is shifted away from the geometric centre of the bounding box, e.g. due to asymmetry or self-shadows.

Each target is also characterised by its status

$$S \in \{new, terminated, updated, occluded, missing\}$$

where *new* is for objects entering the scene or splitting from existing objects, *terminated* for objects exiting the scene or being inactive for some time, *updated* for objects with some measurement, including objects merged or partially occluded, *occluded* for those completely behind static occlusions, and *missing* for those without observation and un-interpreted.

Each *new* object is initialised with the blob features and assigned a zero velocity and a large initial error covariance to encode the uncertainty of the unknown velocity. An *occluded* or *missing* object is updated with the *a priori* estimate with a linearly increasing error covariance. Once the error covariance becomes too large, i.e. the object has no observation for a certain number of frames, a *missing* object becomes a *terminated,* though an *occluded* object will be maintained if the invisible time is less than the expectation (occlusion width / velocity) plus a tolerance reflecting the error covariance.

3.4 Partial Observation

When tracking multiple objects in a complex scene, it is noted that the object measurement, $\mathbf{z}_k$, may be partly unavailable due to dynamic or static occlusion. Fig. 6.4 shows examples of partial observation, where the target is spatially overlapped with another target or a par of the scene. If these partial observations are input into the estimation process during grouping or occlusion, the tracker should be more robust and accurate than those without any observation [17].

We decide the observability of the objects on the basis of the predicted measurement, $\hat{\mathbf{z}}_k^-$, the foreground measurement, $\mathbf{f}_k$, and the scene model. The outcome is represented by the observability vector, $\mathbf{m}_k$, which has the same dimension as the object measurement vector, $\mathbf{z}_k$, with a one-to-one correspondence in their elements. Each element of rrifc has a binary value: *observable* (1) or *unobservable* (0). At the beginning of frame k, each object is set to *observable,* i.e. $\mathbf{m}_k(l) = observable, l \in [1, 6]$, and then subject to a 3-stage modification [18].

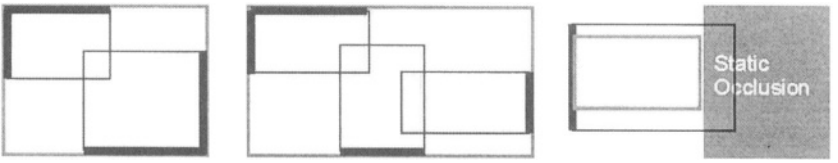

Figure 6.4. Partial observation during dynamic grouping (left and middle) and static occlusion (right).

1 Observability in grouping

For each tracked object and each foreground measurement, a match score is computed on the basis of the Mahalanobis distance, but it is set to zero if the predicted centroid of the object is within the bounding box of the foreground measurement. Each object selects its corresponding foreground measurement in term of minimum match score within a tolerance. For each foreground measurement, the objects that best correspond to it form a group of the objects that are most likely to be merged. The group may include multiple objects, one object, or be empty. For each object in the group, the observability of each bounding edge is modified according to whether this bounding edge is also the corresponding bounding edge of the group (foreground measurement).

2 Observability of foreground measurement

The observability of an object also depends on the observability of its associated foreground measurement. If the bounding box of the underlying foreground region touches the border of the field-of-view, its relevant bounding edge becomes *unobservable* and thus inhibits (masks) the relevant observability for the associated object bounding edge.

3 Observability in static occlusion

When an object is *partly* hidden behind a static occlusion, the measurement of some of its bounding edges become unreliable. For each object, each of the predicted bounding edges is checked. If either corner delimiting that edge is within a static occlusion defined in the scene model, that edge becomes *unobservable.* Once an object is determined as partially occluded by a static occlusion, a bounding box distance measure is used in the data association. It emphasises the minimum distance between corresponding bounding edges, rather than the average distance in the Mahalanobis distance. Therefore, an object may be matched with a much smaller foreground region at the border of a static occlusion.

After the observability of the four bounding edges for i-th object is determined, the observability of its centroid can be decided as *observable* only when all the four bounding edges are *observable.* An alternative but less strict definition considers (r_c, r_1, r_2) and (c_c, c_1, c_2) separately.

For a partially unobservable object, a measurement vector is constituted, whose members can be classified into two inter-correlated blocks (r_c, r_1, r_2) and (c_c, c_1, c_2). The inter-block variables are bound by the constant height (Δr_1 and Δr_2) and constant width (Δc_1 and Δc_2) assumption. Within each block, if all the variables are *unobservable,* the only clue for their measurements are the prediction; if part of its variables is *observable,* the *unobservable* measurements can be deduced from the *observable* measurements. Suppose the observability matrix, $\mathbf{M}_k$, is a diagonal matrix whose main diagonal is the observability vector $\mathbf{m}_k$. The measurement vector is estimated by:

$$\mathbf{z}_k = \mathbf{M}_k\mathbf{f}_k + (\mathbf{I} - \mathbf{M}_k)\mathbf{d}_k \tag{6.6}$$

where $\mathbf{d}_k$ is the deduced measurements of *unobservable* variables from *observable* measurements using some constraints on height and width. The height and width information in the *a priori* state estimate, $\hat{x}_k^-$, is used to compute $\mathbf{d}_k$. When one bounding edge is *unobservable* and its opposite is *observable,* this edge and the centroid are deduced from the opposite edge by assuming Δr_1 and Δr_2 (or Δc_1 and Δc_2) to be constant. When a pair of opposite edges are *observable* but the centroid is *unobservable,* the centroid is deduced by assuming the ratio $\Delta r_1/\Delta r_2$ (or $\Delta c_1/\Delta c_2$) to be constant. If all the variables in an inter-correlated block are *unobservable,* $\mathbf{d}_k = \hat{\mathbf{z}}_k^-$ for that block.

To reflect the lack of actual measurement data the corresponding element in the measurement covariance matrix, $\mathbf{R}_k$, is increased by λ ($\lambda > 1$) to reflect an increased uncertainty. Suppose $\mathbf{R}$ is the measurement covariance matrix for a completely *observable* object, the measurement covariance matrix for a partially observable object becomes:

$$\mathbf{R}_k = \mathbf{M}_k\mathbf{R}\mathbf{M}_k^T + (\mathbf{I} - \mathbf{M}_k)\lambda\mathbf{R}(\mathbf{I} - \mathbf{M}_k)^T \tag{6.7}$$

Because the centroid and bounding edges of each object are measured independently, $\mathbf{R}$ is a diagonal matrix; $\mathbf{M}_k$ is also diagonal, with each element being either 1 or 0. Hence, the equation above can be simplified as:

$$\mathbf{R}_k = [\mathbf{M}_k + \lambda(\mathbf{I} - \mathbf{M}_k)]\mathbf{R} \tag{6.8}$$

Using this equation, an *observable* variable of an object is estimated with a normal measurement variance, whilst an *unobservable* variable is estimated with a larger measurement variance. Therefore, the *observable* bounding edges contribute more to the object estimation than the *unobservable* bounding edges.

Fig. 6.5 shows two examples of object tracking through grouping. In the first example (top row), two previously separate pedestrians (No. 13 and 16) merge and share a large foreground region; then another pedestrian (No. 17) joins in the group, forming a larger foreground region; finally, the group of pedestrian splits. In the second example, a white van (No. 3) first passes by a newly stationary dark car (No. 2), heads toward and occludes a group of people (No. 4), decelerates and stops separately. With the use of partial observation, the position and size of each object are correctly estimated during grouping and all the objects are correctly re-tracked after splitting. To quantitatively evaluate the

performance of this algorithm, we have applied it to the PETS'2001 sequences and compared the results with those using the traditional blind tracking during occlusion. The new algorithm has advantages in terms of 32% decrease in tracking error and 63% improvement in path coherence [17].

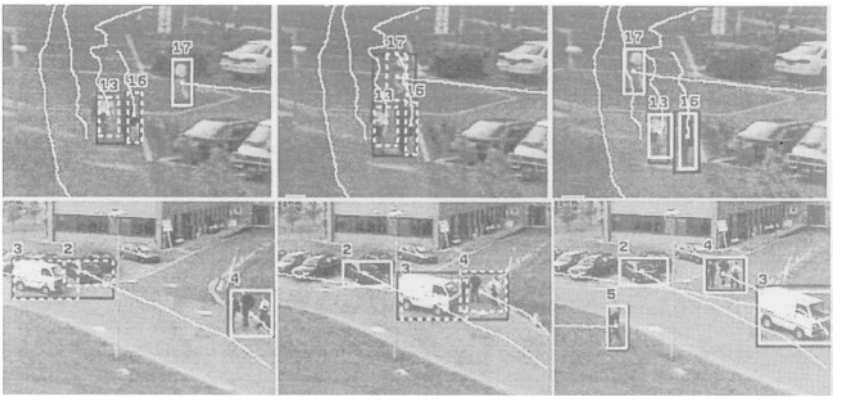

Figure 6.5. Tracking a group of targets with partial observation.

3.5 Target Reasoning

After matching *new* and *updated* objects to blob measurements, there remain some objects that fail to match any blob. This may arise from a variety of reasons: objects leaving the scene; the occlusion of objects by scene elements; the failure of foreground detection, or some unknown reason. The ambiguity can be partly relieved by using domain knowledge. For example, if it is known that the predicted position of an unmatched object overlaps a long-term occlusion, it is most likely that this object left the scene. This is the motivation of a rule-based system. However, there exist uncertainties in such domain knowledge: not all of the objects close to a long-term occlusion will leave the scene — they may walk in front of the occlusion; the foreground detection may fail at any position in a scene; the merging of objects near the occlusion is not reliably detected.

The uncertainties can be encoded in the conditional probabilities between the variables. The object status can then be inferred through a process of deduction. A Bayesian Network [12] is a framework for representing and using domain knowledge to perform probabilistic inference for a set of variables $\mathbf{X} = \{X_1, \ldots, X_N\}$ and has been used in motion analysis in [3]. A BN is a directed acyclic graph in which nodes represent random variables and arcs represent causal connections among the variables. Associated with each node is a conditional probability table given each possible state of its parents. In the case that a node has no parents, conditional probabilities degenerate to priors. By assuming conditional independence between some variables, the joint probability distribution for variable $\mathbf{X}$ to have the value, $\mathbf{x} = \{x_1, \ldots, x_N\}$, is

given by:

$$P(\mathbf{x}) = \prod_{i=1}^{N} P(x_i|pa(x_i)) \tag{6.9}$$

When evidence **e** are observed for a subset of the nodes, the estimate for any of the remaining nodes can be computed to maximise the posterior probabilities:

$$\hat{\mathbf{x}} = \text{argmax}_{\mathbf{x}} P(\mathbf{x}|\mathbf{e}) \tag{6.10}$$

The Bayesian network used for reasoning about unmatched objects is shown in Fig. 6.6. The variables EB ("Exit from BO"), EL ("Exit from LO"), OS ("Occluded by SO"), and MS ("Missing") are query nodes. The distance measures D(BO), D(LO) and D(SO), as well as the variable UM ("Unmatched object"), are evidence nodes. To simplify the representation and learning of the conditional probabilities, all quantities in the network are discretized into binary values. The distance measures used are the bounding box distance and include:

- D(BO) — the distance to the BO. D(BO) = 0 if the predicted bounding box of an object is completely inside the BO.
- D(LO) — the distance to the closest LO. D(LO) = 0 if the predicted bounding box of an object is inside an LO.
- D(SO) — the distance to the closest SO. D(SO) = 0 if the predicted bounding box of an object is inside an SO.

The network structure is determined using domain knowledge: UM ("Unmatched object") usually arises from an object exiting from the border occlusion (EB), exiting from a long-term occlusion (EL), occluded by a short-term occlusion (OS), or missing for unknown reasons (MS); when EB ("Exit from BO") is true, the prediction of the underlying object usually goes into the border occlusion so that D(BO) is small; when EL ("Exit from LO") is true, the prediction of the underlying object tends to overlay a long-term occlusion so that D(LO) is small; when OS ("Occluded by SO") is true, the prediction of the underlying object is often very close to a short-term occlusion so that D(SO) is small. The prior and conditional probabilities attributed to the variables in the network are listed as follows. To reduce the number of conditional probabilities to specify for node UM, the noisy-OR model is applied among its four parents EB, EL, OS and MS. This model assumes that (1) each parent has an independent chance of causing UM, (2) all the possible causes are listed (node MS works as the leak node explaining miscellaneous causes; once MS is true, it always gives rise to an unmatched object, i,e. $\mathrm{P(UM = T \mid MS = T) = 1)}$, and (3) whatever inhibits a parent from causing UM = T is independent of whatever inhibits another parent from causing UM = T. The independent inhibitor probabilities are $q_1 = \mathrm{P(UM = F \mid EB = T)}$, $q_2 = \mathrm{P(UM = F \mid EL = T)}$, $q_3 = \mathrm{P(UM = F \mid OS = T)}$, and the conditional probability table for node UM is as follows:

By using noisy-OR relationships, the variable UM, which depends on $n = 4$ parents, can be described using $O(n)$ parameters instead of $O(2^n)$ for the full

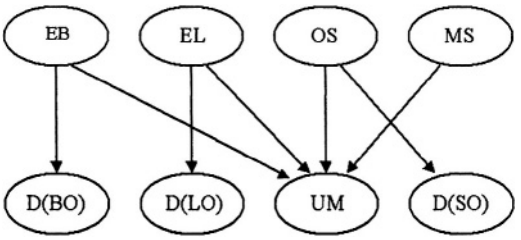

Figure 6.6. The Bayesian network for reasoning about unmatched objects.

Table 6.1. Conditional probabilities in Bayesian network.

EB	EL	OS	P(UM=F)	P(UM=T)
F	F	F	$1-p_4$	p_4
T	F	F	$q_1(1-p_4)$	$1-q_1(1-p_4)$
F	T	F	$q_2(1-p_4)$	$1-q_2(1-p_4)$
T	T	F	$q_1q_2(1-p_4)$	$1-q_1q_2(1-p_4)$
F	F	T	$q_3(1-p_4)$	$1-q_3(1-p_4)$
T	F	T	$q_1q_3(1-p_4)$	$1-q_1q_3(1-p_4)$
F	T	T	$q_2q_3(1-p_4)$	$1-q_2q_3(1-p_4)$
T	T	T	$q_1q_2q_3(1-p_4)$	$1-q_1q_2q_3(1-p_4)$

conditional probability table. The entire network is specified by 13 parameters, including 4 priors, 3 inhibitor probabilities and 6 conditional probabilities. Given the observed values for the evidence nodes, the posterior probabilities of any unmatched object caused by EB, EL, OS or MS are computed using the junction tree algorithm and the most probable explanation can be given.

These prior and conditional probabilities can be learned automatically over sample sequences taken at the same scene. For the four query nodes, this is to estimate the probability $P(x_i = T) = \theta_i$, given a set D of N observations. Suppose the likelihood to generate data D is a binomial distribution $P(D|\theta_i) = C(N, N_T)\theta_i^{N_T}(1-\theta_i)^{N-N_T}$ where $N_T = N(x_i = T)$ and θ_i has a Beta prior $P(\theta_i) = Beta(\alpha_T, \alpha - \alpha_T)\theta_i^{\alpha_T - 1}(1-\theta_i)^{\alpha-\alpha_T-1}$. The maximum *a posteriori* (MAP) estimate [9] for θ_i is:

$$\hat{\theta_i^{MAP}} = \frac{\alpha_T + N_T}{\alpha + N} \tag{6.11}$$

The hyperparameters α and α_T can be thought of as imaginary counts from our prior experience, equivalent to a sample size α . For the four evidence nodes the MAP estimate is:

$$\theta^{M\hat{A}P}_{x_i|pa(x_i)} = \frac{\alpha_T(x_i, pa(x_i)) + N_T(x_i, pa(x_i))}{\alpha(pa(x_i)) + N(pa(x_i))} \tag{6.12}$$

To estimate the state of the query nodes in learning probabilities, a decide-and-verify strategy is used on the assumption that foreground detection failures are very unlikely to occur and are independent. The EB or EL is thought of as true if an object with its prediction overlaying the BO or an LO cannot find an associated foreground region over a certain number of frames. The OS is believed true if an object with its prediction overlaying an SO cannot find an match over the expected invisible time plus a tolerance. The MS is considered true if an object cannot find a match over a certain number of frames for other reasons not trivial to automatically recognise. These reasons include foreground detection failure (targets moving beyond detection range, long-stationary targets absorbed into background, targets walking in front of a scene element with similar colour) and failure to build some LOs and SOs that actually exist in the scene. Therefore, an incomplete construction of the scene model will lead to a greater prior for the MS node. In addition, it is the local conditional probabilities for each LO or SO, rather than the global ones for all LOs or all SOs, that need to be learned. Such a local conditional probability represents how likely objects are to go behind a specific occlusion, while the EL's and OS's priors indicate the global occlusion density in the scene. When reasoning about the status of an object, the local conditional probabilities used are those of the nearest LO and SO to the object. Fig. 6.7 shows the tracking of multiple

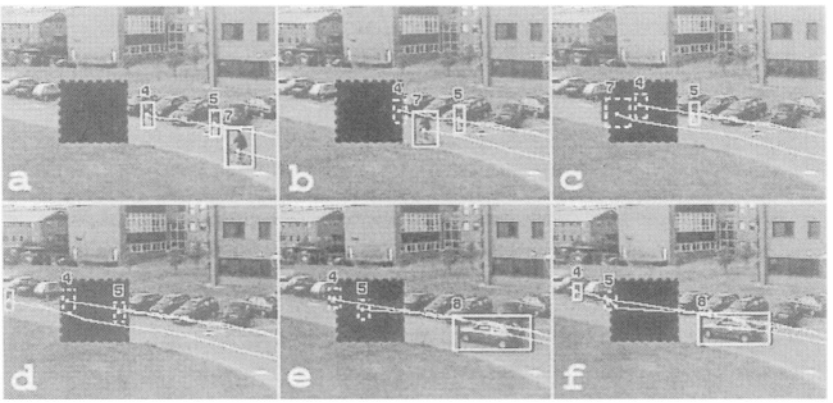

Figure 6.7. Tracking of objects completely behind a static occlusion.

targets completely occluded by a synthetic occlusion. In Figs. 6.7(b) and (c), when a pedestrian (No. 4) and a cyclist (No. 7) disappear behind the occlusion, they are classified as occluded objects by the target reasoning stage and their location estimate is predicted by the tracker. If their predicted emergence from the opposite side of the occlusion is delayed (i.e. the expected reappearance time elapses) but no match is found to any foreground region, these targets are

"held" at the border of the occlusion for a certain number of frames to cover the error covariance while anticipating a match (Figs. 6.7(c) and (d)). In this case, our algorithm correctly tracks each of the three targets. In all of the 23 objects passing behind the black occlusion added to the PETS'2001 sequences, only one object is incorrectly re-tracked (when two targets reappear at the same time), while a traditional algorithm without sensing the presence of any static occlusion reports three failures and a 59% increase in tracking error.

4. Multi view tracking

The task of multi view object tracking is comprised of many tasks. Initially, moving objects of interest must be identified in each camera. This represents a challenging problem, particularly in outdoor environments where lighting conditions cannot be controlled and image intensities are subject to large changes in illumination variation. Each camera in the surveillance network has an intelligent sensor, which employs a robust motion segmentation and object tracking strategy as was discussed in the previous section. It is assumed that each camera view is fixed and calibrated in a world coordinate system. The multi view object tracking framework should be able to integrate tracking information from each camera and reliably track objects between views. In addition the multi view tracker should be able to resolve both dynamic occlusions that occur due to object interaction, and static occlusions that can occur due to the scene constraints, for example trees that form occlusion regions.

The multi view tracker uses a training phase to learn information about the scene, which can facilitate the integration and object tracking process. This can include learning the relations between each view to allow feature correspondence to assign a unique label for an object even it is visible in several views simultaneously. In a typical surveillance environment the cameras are placed to maximise the field of coverage of the scene. As a consequence some cameras will have limited overlap, which increases the difficulty of tracking objects without loss of identity. Hence, the multi view tracker must be able to track objects between non-overlapping and spatially adjacent views. The system should be able to exploit spatial cues to maintain the identity of tracked objects. Since the object disappears from the field of view temporarily, to increase the likelihood of matching the object on reappearance it will be necessary to record attributes of the object at the time of its exit.

4.1 Homography Estimation

A homography defines a planar mapping between two camera views that have a degree of overlap. The homography mapping provides a mechanism for matching object features between overlapping camera views. Most surveillance scenes conform to the ground plane constraint, allowing the homography to be applied for matching objects between different camera views. The application of the homography is illustrated in Fig. 6.8 where it is used to correspond objects between three overlapping camera views. The black arrows show how

the homography can project the centroid of objects on the same ground plane between different views.

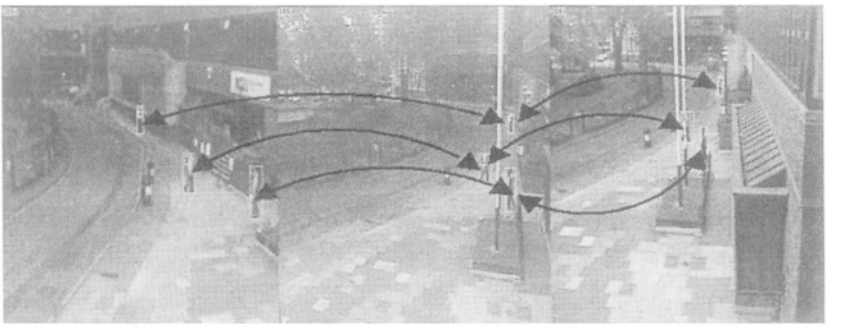

Figure 6.8. Example of viewpoint correspondence between three overlapping camera views.

4.2 Least Median of Squares

Given a set of correspondence points the homography can be estimated using the method described in section 7.2. The next step is to define a process that would allow a set of correspondence points to be determined automatically from a set of input data. Given a set of sparse object trajectories they can be used to provide training data for estimating the homography transformation between the two overlapping camera views. The object trajectories are taken during periods of low activity, in order to reduce the likelihood of finding false correspondence points. The trajectories take the form of a set of tracked object centroids that are found using the feature detection method described in section 3.3. A Least Median of Squares (LMS) approach is used to automatically recover a set of correspondence points between each pair of overlapping camera views, which can be used to compute the homography mapping. The LMS method performs an iterative search of a solution space by randomly selecting a minimal set of correspondence points to compute a homography mapping. The solution found to be the most consistent the set of object trajectories is taken as the final solution. Stein [14] used this method for registering ground planes between overlapping camera views. The LMS method was used since it is a robust alignment algorithm where the data contains a number of outliers

The homography relations between each overlapping camera are used to match detected moving objects in each overlapping camera view. The transfer error is the summation of the projection error in each camera view for a pair of correspondence points. It indicates the size of the error between corresponded features and their expected projections according to some translating function, the object centroid homography in our case. In Mikic, et al. [11] the 3D epipolar constraints were used as a basis for matching. Each method has its own advantages. The epipole line based approach can still function even if the two views do not share a common dominant ground plane but requires that

the camera geometry between the two views is known in advance and fairly accurate. The homography-based method assumes that each camera view shares a dominant ground plane.

The biggest advantage of the homographic method over the epipole based method is that the homography maps points to points, while the epipole approach maps points to lines, so a one dimensional search still needs to be performed to establish an object correspondence. The homographic method could be applied to all regions ofthe image assuming that we had 3D camera geometry along with terrain information of the scene, for example an elevation map. A graphical depiction is given for the feature matching in Fig. 6.9. The bounding box of each object is displayed. The white lines represent the epipole lines for each object centroid terminating at the ground plane. The black points represent the tracked centroid of each detected object. The white points in each bounding box represent the projection of the object centroid using the homography as a transformation. The two images in Fig. 6.9 shows an example of matching two vehicles The transfer error is used by the homography alignment and viewpoint correspondence methods for assessing the quality of a corresponded pair of centroids in two different camera views. The transfer error associated with a correspondence pair is defined as:

$$d(x', H^{-1}x'')^2 + d(x'', Hx')^2$$

where x' and x'' are projective coordinates in view 1 and view 2 respectively, H is the homography transformation from view 1 to view 2, and $d(a, b)$ is the Euclidean distance between a pair of projective coordinates a and b.

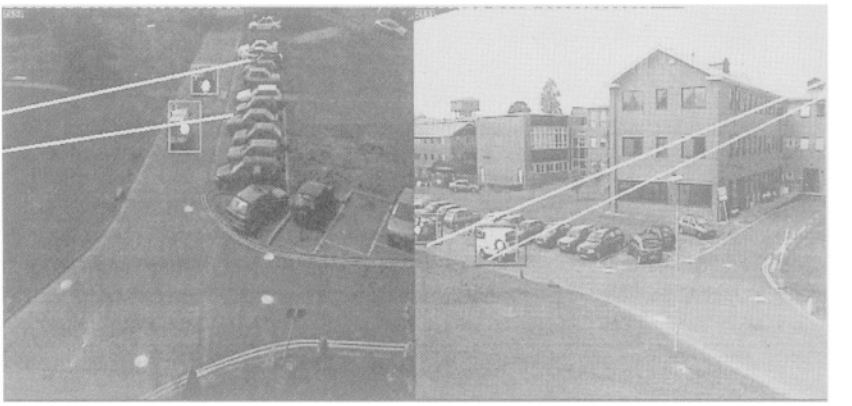

Figure 6.9. Feature matching using: epipole line analysis and homography alignment.

4.3 Feature Matching Between Overlapping Views

The LMS algorithm described in section 4.2 is used to determine a set of correspondence points, which were then used to compute the object centroid

homography. This homography can be used to correspond object tracks in the testing video sequences. Once the calibration data and homography alignment model are available we can use the relationship between both camera views to correspond detected objects. From observing the results of motion detection it is apparent that the object centroid is a more stable feature to track in 3D, since it is more reliably detected than the top or bottom of the object, particularly in outdoor scenes where the object may be a far distance from the camera.

To summarize the following steps are used for matching 2D object tracks, taken from different views, for a given image frame:

1 Create a list of all possible correspondence pairs of objects for each camera view.

2 Compute the transfer error for each object pair

3 Sort the correspondence points list by increasing transfer error.

4 Select the most likely correspondence pairs according to the transfer error. Apply a threshold so that correspondence pairs where Transfer Error$> \epsilon_{max}$ are not considered as potential matches.

5 Create a correspondence points list for each matching object

6 Map each entry in the correspondence points list using 3D line intersection of the bundle of image rays to locate the object in 3D.

7 For each object centroid which does not have a match in the correspondence pair list use the calibration information to estimate the location of the object in 3D

Two additional constraints are applied to the viewpoint correspondence is that we do not allow one to many mappings between observations in each camera view. This has the effect of reducing the number 'phantom' objects which can appear at the end of a dynamic occlusion. An example of viewpoint correspondence is shown in Fig. 6.10, the left image shows the original objects detected by the 2D object tracker, and the right image shows the observations remaining once viewpoint correspondence has been applied. It can be observed that the number of phantom objects near the lamppost in left camera view have been eliminated by the feature matching process. The viewpoint correspondence process has the affect of reducing the number of false objects that have been detected by the 2D object tracker.

4.4 3D Measurements

Given a set of corresponded object features along and camera calibration information it is possible to extract 3D measurements from the scene. Using multiple viewpoints improves the estimation of the 3D measurement. A 3D line intersection algorithm is employed to estimate each object's location in world coordinates. Using the calibrated camera parameters it is also possible to

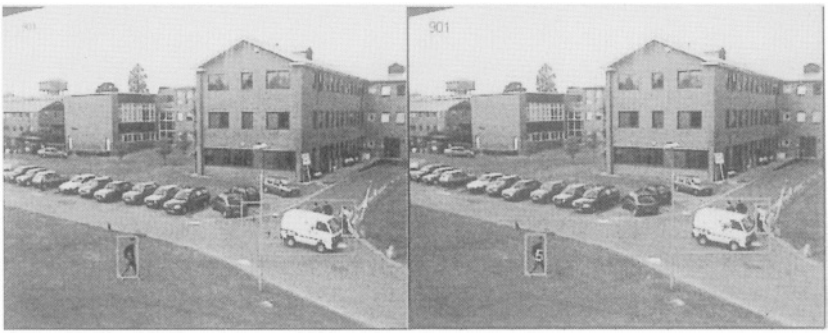

Figure 6.10. Example of feature matching between overlapping views.

estimate the uncertainty of the image measurement by propagating the covariance from the 2D image plane to 3D world coordinates. A 3D line intersection algorithm is used to estimate the location of an object in a least squares sense. Details of the method used to perform this process is given in section 7.3.

4.5 Tracking in 3D

Using the approach described in sections 4.2 and 4.3 we are able to merge the object tracks from separate camera views into a global world coordinate view. This 3D track data provides the set of observations, which are used for tracking using the Kalman filter.

The Kalman filter provides an efficient recursive solution for tracking the state of a discrete time controlled process. The filter has been applied in numerous tracking applications for visual surveillance. The 3D Kalman filter tracker assumes a constant velocity model. A summary is given of the state model used for tracking in following equations. At each state update step the observation covariance is set according to the measurement uncertainty determined by projecting a nominal image covariance from the image to the 3D object space.

State Model

$\mathbf{X}_t = [\ x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z}\ ]^T$ where, (x, y, z) is the spatial location in world coordinates, and $(\dot{x}, \dot{y}, \dot{z})$ is the spatial velocity in world coordinates.

State Transition Model **Observation Model**

$$A = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (6.13)$$

Where T is the difference of the time of capture of the current and previous image frame. The Kalman filter allows the system to reliably track the object state even if the objects disappear from the camera view for a five frames. This value was set based on empirical evidence.

3D Data Association. The 2D object tracks detected by the background subtraction are converted to a set of 3D observations using the 3D line intersection algorithm as discussed in section 4.3. Since the system is tracking multiple objects in 3D it is necessary to ensure that each observation is assigned to the correct object being tracked. This problem is generally referred to as the data association problem. The Mahalanobis distance provides a probabilistic solution to find the best matches between the predicted states of the tracked objects and the observations made by the system:

$$M_D = (H\hat{X}_k^- - Z_k)^T (H\hat{P}_k^- H^T + R_k)^{-1} (H\hat{X}_k^- - Z_k) \qquad (6.14)$$

An appropriate threshold can be chosen for the Mahalanobis distance by selecting a value that gives a 95% confidence for a match, assuming a chi-square distribution. The dimension of the observations for the 3D tracker is 3, hence the value threshold selected for M_D was 7.81. A Mahalanobis distance table is created between each tracked object and observation. The system then assigns each observation to each tracked object based upon the size of the Mahalanobis distance.

Outline of 3D Tracking Algorithm. The following is a summary of the steps used to update each tracked 3D object for a giving image frame:

1 For each 3D observation in frame T, create a Mahalanobis distance table and sort by the distance measure. Threshold the values such that each Mahalanobis distance $< \xi_{max}$

2 For each existing tracked object:

   (a) Select the observation which has the largest likelihood of being a match

   (b) Update the tracked object using this observation

3 For each existing tracked object not matched to an observation

   (a) If the tracked object has not been matched in K frames then mark it as deleted, else

   (b) Update the tracked object using the predicted state estimate

4 For each unmatched observation:

   (a) Create a new tracked object, using the observation to initialise the object state

   (b) Set the initial covariance of the object state.

The following two constraints are applied during the object state update process:

1 Each observation can be used to update only one existing tracked object.

2 A new tracked object can only be created when its initial observation does not match an existing tracked object

The first constraint prevents an object being updated during a dynamic occlusion when the observation is not consistent with its predicted trajectory. The second constraint prevents new tracks being created at the end of a dynamic occlusion when the objects separate.

Multi View Tracking Example. Fig. 6.11 shows how dynamic occlusions can be resolved in 3D when tracking objects between two camera views. The left figure shows a pedestrian being occluded by a vehicle, while the right image shows a group of pedestrians being occluded by a white van. The 3D ground plane map shows that the tracked objects can still be tracked in 3D during the dynamic occlusions that occur in both camera views.

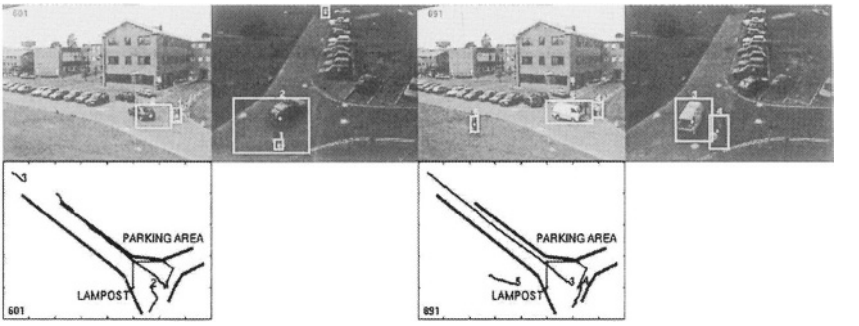

Figure 6.11. Examples of handling dynamic occlusion.

4.6 Non-Overlapping Views

In a typical image surveillance network the cameras are usually organised so as to maximise the total field of coverage. As a consequence there can be several cameras in the surveillance network that are separated by a short temporal and spatial distance, or have minimal overlap. In these situations the system needs to track an object when it leaves the field of view of one camera and the re-enters the field of view of another after a short temporal delay. For short time durations of less than two seconds the trajectory prediction of the Kalman filter can be used to predict where the object should become visible again to the system. However, if the object changes direction significantly of or disappears for a longer time period this approach is unreliable. In order to handle these cases the system uses an object handover policy between the pair

of non-overlapping cameras. The object handover policy attempts to resolve the handover of objects that move between non-overlapping camera views. The system waits for a new object to be created in the adjacent camera view. A data association method is applied to check the temporal constraints of the objects exit and re-entry into the network field of view.

Object Handover Regions. The object handover region models consist of a linked entry and exit region along with the temporal delay between each region. The major entry and exit regions in each camera are automatically learned by post-track analysis of tracking data. Given the set of entry and exit regions the learning component of the surveillance system also determines the spatial links between entry and exit zones in different camera views, as discussed in depth in Chapter 7. The temporal delay can be determined manually by observation, or by generating statistics from the data stored in the database. The temporal delay gives an indication of the transit time for the handover region for a specific object class, so the temporal delay for a pedestrian object class and a vehicle object class would be different based on the set of observations used to generate the statistics. Each entry and exit region is modelled as a Gaussian:

$$\langle (x, y), \Sigma \rangle \tag{6.15}$$

where (x, y) is the centre of the distribution in 2D image coordinates, and Σ is the spatial covariance of the distribution. The following convention is used to describe the major entry and exit regions in each camera view:

- X_i^k is the k-th exit region in the i-th camera view.
- E_j^l is the l-th entry region in the j-th camera view.

Given the set of major exit and entry regions in each camera the following convention is used to define the handover regions between the non-overlapping camera views: $H_{ij}^p = \langle X_i^k, E_j^l, t, \sigma \rangle$ is the p-th handover region between camera i-th and j-th camera views.

As previously discussed each handover region H_{ij}^p consists of a spatially connected exit and entry region pair (X_i^k, E_j^l), along with the temporal delay and the variance of the temporal delay (t, σ) between the exit and entry region. An example of object handover regions is visually depicted in Fig. 6.12. The black and white ellipses in each camera view represent the major entry and exit regions in each camera. The links represent the handover regions between each camera.

Object Handover Agents. The object handover mechanism only needs to be activated when an object is terminated within an exit region that is linked to an entry region in the adjacent camera view. Once the object leaves the network field of view and is in transit between the non-overlapping views the system cannot reliably track the object.

Handover Initiation

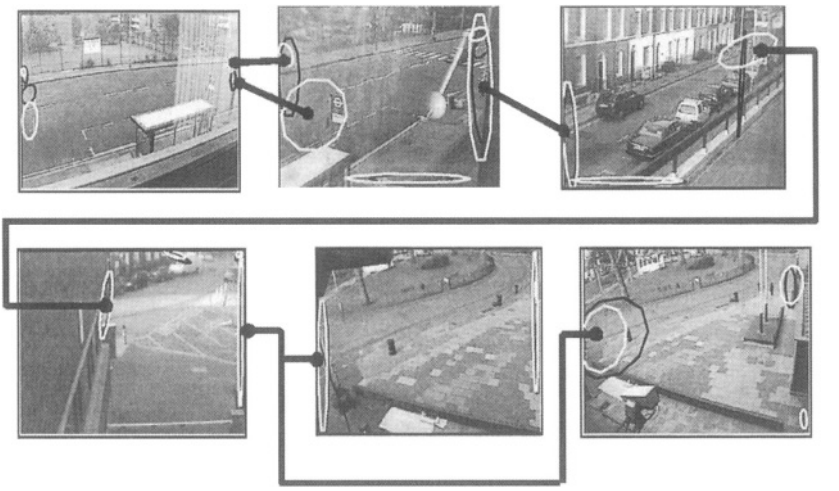

Figure 6.12. Handover regions for six cameras in the surveillance system.

The handover agent is activated when an object is terminated within an exit region X_i^k that is included in the handover region list. The handover agent records the geometric location, and time when the object left the field of view of the ith camera. Allowing the object handover agent to only be activated when an object is terminated in a handover region eliminates the case where an object is prematurely terminated within the field of view due to tracking failure caused by a complex dynamic occlusion. In addition, once the handover agent has been activated the handover region model can be used to determine the most likely regions where the object is expected to re-appear, hence reducing the computational cost of completing the handover process.

Handover Completion

The handover agent achieves completion when an object is created within the entry region E_j^l that forms a handover region with the exit region X_i^k, where the object was terminated in the ith camera view. The handover agent task is only complete if the new object satisfies the temporal constraints of the handover region. The new object location must be consistent with the temporal delay of the handover region and the transit time of when the object left and reappeared in the ith and jth camera view respectively.

Handover Termination

The handover agent is terminated once an object has not been matched after a maximal temporal delay, which can be determined by the statistical properties of the handover regions related to the exit region where the object left the field of view. The maximal temporal delay in a handover region is an important characteristic, since the surveillance regions are not constrained in such a way that an object must re-appear in the field of view once it enters a handover

region. It is possible that once the object has been terminated within an exit region it will not re-appear within the network field of view. When this case occurs it is not possible for the system to locate the object, since it is will not be visible by any of the cameras in the surveillance network.

The framework used for tracking object between non-overlapping views makes several assumptions. It is assumed that the temporal delay between the camera views is of the order of seconds for each object class. If the handover regions are located on the same ground plane and calibrated in the same world coordinate system then 3D trajectory prediction can be used to add another constraint to the data association between the handover object and candidate objects which appear in entry regions in the adjacent camera view. The 3D trajectory prediction is only valid if the object maintains the same velocity and does not significantly change direction once it has entered the handover region.

An example of the object handover reasoning process is given in Fig. 6.13. The identity of the vehicle is correctly preserved as it moves through four of the cameras in the surveillance network. The black lines represent the handover regions used to coordinate the tracking of objects between each of the views. The arrows on each line indicate the direction of the vehicle's motion between the camera views.

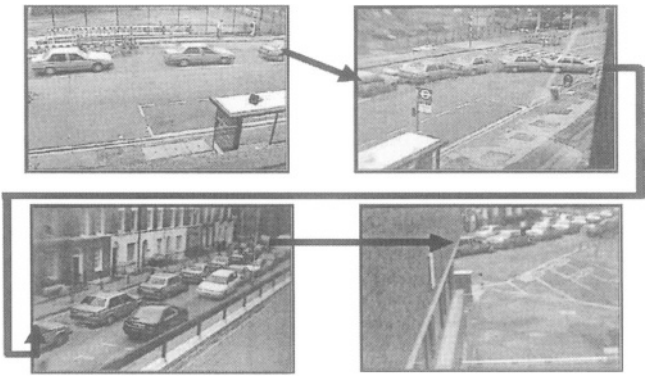

Figure 6.13. Object handover between four camera views.

5. System architecture

The surveillance system employs a centralised control strategy as was described in the introduction of this chapter. The multi view tracking server (MTS) creates separate receiver threads (RT) to process the data transmitted by each intelligent camera unit (ICU) connected to the surveillance network. Each ICU transmits tracking data to each RT in the form of symbolic packets. The system uses TCP/IP sockets to exchange data between each ICU and RT. Once the object tracking information has been received it is loaded into the surveillance

database that can be accessed for subsequent online or offline processing. Each ICU and RT exchanges data using the following four message types:

Background Image Message

This message format is used to transmit the background image from the ICU to the MTS during the initial start-up of the camera. The MTS uses the background image to visualise the tracking activity in 2D. The background image can be periodically refreshed to reflect changes in the camera viewpoint.

Timestamp Message

This message is used to transmit the timestamp of the current image frame processed by the ICU to the MTS. The MTS uses the recorded time-stamps to perform temporal alignment

2D Object Track Message

This message is generated for each detected object in the current image frame. The message includes details of the object's location, bounding box dimensions, and normalised colour components.

2D Object Framelet Message

Each detected object in the current frame is extracted from the image and transmitted to the MTS. The framelet is then stored in the surveillance database and can be plotted on the background image to visualise the activity within the field of view of the ICU.

5.1 Surveillance Database

The key design consideration for the surveillance database was that it should be possible to support a range of low-level and high-level queries. At the lowest level it is necessary to access the raw video data in order to observe some object activity recorded by the system. At the highest level a user would execute database queries to identify various types of object activity observed by the system. In order to support these real-time operational and reporting requirements we use a multi-layered database design, where each layer represents a different abstraction of the original video data. The surveillance database comprises three layers of abstraction:

- Image framelet layer
- Object motion layer
- Semantic description layer

This three-layer hierarchy supports the requirements for real-time capture and storage of detected moving objects at the lowest level, to the online query of activity analysis at the highest level. Computer vision algorithms are employed to automatically acquire the information at each level of abstraction. The physical database is implemented using PostgreSQL running on a Linux server. PostgreSQL provides support for storing each detected object in the database. This provides an efficient mechanism for real-time storage of each object detected by the surveillance system. In addition to providing fast indexing and retrieval of data the surveillance database can be customised to offer remote access via a graphical user interface and also log each query submitted

by each user. For the remainder of this chapter we focus on the image framelet and object motion layers of the surveillance database. The semantic description layer is described in the visual surveillance learning chapter.

Image Framelet Layer. The image framelet layer is the lowest level of representation of the raw pixels identified as a moving object by each camera in the surveillance network. Each camera view is fixed and background subtraction is employed to detect moving objects of interest. The raw image pixels identified as foreground objects are transmitted via a TCP/IP socket connection to the surveillance database for storage This MPEG-4 like coding strategy enables considerable savings in disk space, and allows efficient management of the video data. Typically, twenty-four hours of video data from six cameras can be condensed into only a few gigabytes of data. This compares to an uncompressed volume of approximately 4 terabytes for one day of video data in the current format we are using, representing a compression ratio of more than 1000:1.

In Figure 6.14 an example is shown of some objects stored in the image framelet layer. The images show the motion of two pedestrians as they move through the field of view of the camera. Information stored in the image framelet layer can be used to reconstruct the video sequence by plotting the framelets onto a background image.

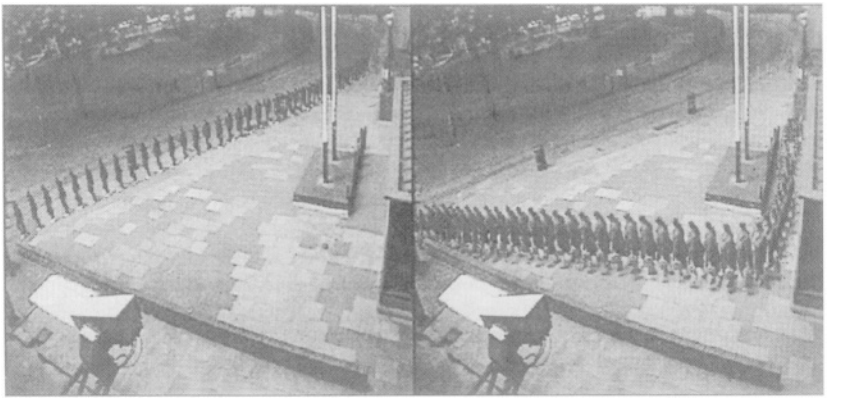

Figure 6.14. Example of objects stored in the image framelet layer.

The main attributes stored in the image framelet layer are described in Table 6.2. An entry in the image framelet layer is created for each object detected by the system. It should be noted that additional information, such as the time when the object was detected is stored in other underlying database tables. The raw image pixels associated with each detected object is stored internally in the database. The PostgreSQL database compresses the framelet data, which has the benefit of conserving disk space.

Table 6.2. Attributes stored in image framelet layer.

Field Name	*Description*
Camera	The camera view
Videoseq	The identification of the captured video sequence
Frame	The frame where the object was detected
Trackid	The track number of the detected object
Bounding_box	The bounding box describing the region where the object was detected
Data	A reference to the raw image pixels of the detected object

Object Motion Layer. The object motion layer is the second level in the hierarchy of abstraction. Each intelligent camera in the surveillance network employs a robust 2D tracking algorithm to record an object's movement within the field of view of each camera. Features are extracted from each object including: bounding box, normalized colour components, object centroid, and the object pixel velocity. Information is integrated between cameras in the surveillance network by employing a 3D multi view object tracker, which tracks objects between partially overlapping, and non-overlapping camera views separated by a short spatial distance. Objects in overlapping views are matched using the ground plane constraint. A first order 3D Kalman filter is used to track the location and dynamic properties of each moving object, which was discussed in section 4.

The 2D and 3D object tracking results are stored in the object motion layer of the surveillance database. The object motion layer can be accessed to execute offline learning processes that can augment the object tracking process. For example, a set of 2D object trajectories can be used to automatically recover the homography relations between each pair of over-lapping cameras, as was discussed in chapter 7. The multi view object tracker robustly matches objects between overlapping views by using these homography relations. The object motion and image framelet layer can also be combined in order to review the quality of the object tracking in both 2D and 3D. The key attributes stored in the object motion layer are described in Table 6.3 and Table 6.4.

In Fig. 6.15 results from both the 2D tracking and multi-view object tracker are illustrated. The six images represent the viewpoints of each camera in the surveillance network. Cameras 1 and 2, 3 and 4, and 5 and 6 have partially overlapping fields of view. It can be observed that the multi-view tracker has assigned the same identity to each object. Fig. 6.16 shows the field of view of each camera plotted onto a common ground plane generated from a landmark-based camera calibration. 3D motion trajectories are also plotted on this map in order to allow the object activity to be visualized over of the entire surveillance region.

6. Summary

We have considered in this chapter a number of the computer vision tasks that will underpin the operation of an automated visual surveillance system.

Table 6.3. Attributes stored in object motion layer (2D Tracker).

Field Name	*Description*
Camera	The camera view
Videoseq	The identification of the captured video sequence
Frame	The frame where the object was detected
Trackid	The track number of the detected object
Bounding_box	The bounding box describing the tracked region of the object
Position	The 2D location of the object in the image
Appearance	The normalized colour components of the tracked object

Table 6.4. Attributes stored in object motion layer (Multi View Tracker).

Field Name	*Description*
Multivideoseq	The identification of the captured multi video sequence
Frame	The frame where the object was detected
Trackid	The track number of the detected object
Position	The 3D location of the tracked object in ground plane coordinates
Velocity	The velocity of the object

Figure 6.15. Camera network on University campus showing 6 cameras distributed around the building, numbered 1-6 from top left to bottom right, raster-scan fashion.

The requirements placed on a real surveillance system are severe, with the need to operate over a wide range of varying illumination and weather conditions, whilst coping with the complexity of the perceptual challenges presented within the region under observation. Multiple cameras are prerequisite for most surveillance installations, and the system must be able to integrate the data de-

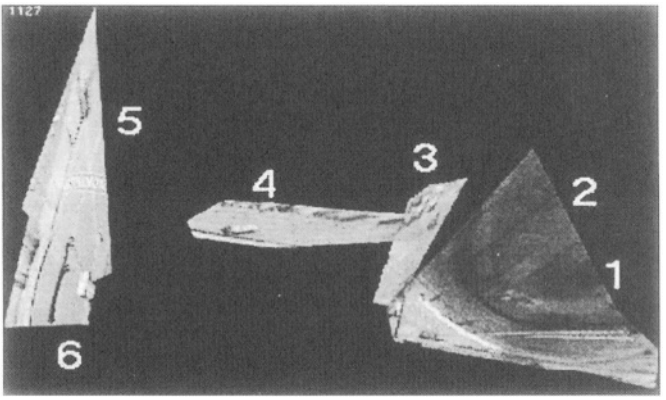

Figure 6.16. Reprojection of the camera views from Fig. 6.15 onto a common ground plane, showing tracked objects trajectories plotted into the views (white and black trails).

rived from all the cameras in order to best exploit their information gathering capabilities. Our approach is based on algorithms that can adapt to the changing conditions to ensure robust tracking of objects, minimising the effects of occlusion that would otherwise severely limit this robustness. In addition, we wish to minimise the amount of location-specific information that must be provided to the system, instead preferring to learn this information by observing the scene(s) and extracting geometric and semantic information that is used to construct models of the environment. These models are employed to augment the spatio-temporal reasoning of the tracking algorithms and will be essential aids for performing the subsequent stages of activity and behavioural analysis.

7. Appendix

7.1 Kalman Filter

A Kalman filter based on a first-order motion model is used to track each object according to the object measurement vector. The state transition and measurement equations are:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \quad (6.16)$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \quad (6.17)$$

where $\mathbf{w}_k$ and $\mathbf{v}_k$ are process noise and measurement noise, respectively, and $\mathbf{w}_k \sim N(\mathbf{0}, \mathbf{Q}_k)$, $\mathbf{v}_k \sim N(\mathbf{0}, \mathbf{R}_k)$. $\mathbf{A}$ is the state transition matrix, and $\mathbf{H}$ is the measurement matrix. The *a priori* estimate $\hat{\mathbf{x}}_k^-$, the *a posteriori* estimate

$\hat{\mathbf{x}}_k^+$ and the predicted measurement $\hat{\mathbf{z}}_k^-$ are iteratively computed by:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\mathbf{x}_{k-1}^+ \tag{6.18}$$

$$\hat{\mathbf{z}}_k^- = \mathbf{H}\hat{\mathbf{x}}_k^- \tag{6.19}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k^-) \tag{6.20}$$

where $\mathbf{K}_k$ is the Kalman gain matrix that is sought to minimise the *a posteriori* error covariance $\mathbf{P}_k^+$ in a least-square sense and can also be computed with $\mathbf{P}_k^+$ and the *a priori* error covariance $\mathbf{P}_k^-$ iteratively:

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}^+\mathbf{A}^T + \mathbf{Q}_{k-1} \tag{6.21}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T[\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}_k]^{-1} \tag{6.22}$$

$$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{K}_k\mathbf{H}]\mathbf{P}_k^- \tag{6.23}$$

7.2 Homography Estimation

A homography mapping defines a planar mapping between two overlapping camera views:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{6.24}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{6.25}$$

Where (x, y) and (x', y') are image coordinates for the first and second camera views respectively. Hence, each correspondence point between two camera views results in two equations in terms of the coefficients of the homography. Given at least four correspondence points allows the homography to be evaluated. It is most common to use Singular Value Decomposition (SVD) for computing the homography. The homography matrix can be written in vector form:

$$\mathbf{H} = [\ h_{11}\ \ h_{12}\ \ h_{13}\ \ h_{21}\ \ h_{22}\ \ h_{23}\ \ h_{31}\ \ h_{32}\ \ h_{33}\ ]^T \tag{6.26}$$

Each pair of correspondence points, (x, y) and (x', y'), results in two equations in terms of the coefficients of the homography matrix:

$$[\ x_i\ \ y_i\ \ 1\ \ 0\ \ 0\ \ 0\ \ -x_ix_i'\ \ -yx_i'\ \ -x_i'\ ]\mathbf{H} = 0 \tag{6.27}$$

$$[\ 0\ \ 0\ \ 0\ \ x_i\ \ y_i\ \ 1\ \ -xy_i'\ \ -yy_i'\ \ -y_i'\ ]\mathbf{H} = 0 \tag{6.28}$$

Given N correspondence points a $(2N \times 9)$ matrix M that can be constructed and then used to minimise $||M\mathbf{H}||$ subject to the constraint $||\mathbf{H} = 1||$. The value of the homography matrix can then be estimated by using Singular Value Decomposition.

7.3 3D Measurement Estimation

Given a set of corresponded objects in each camera view a 3D ray is projected through the centroid of the object in order to estimate its location. Using the camera calibration model it is possible to map the 2D object centroid to a 3D line in world coordinates.

Given a set of N 3D lines $\mathbf{r}_i = \mathbf{a}_i + \lambda_i \mathbf{b}_i$ a point $\mathbf{p} = (x, y, z)^T$ must be evaluated which minimises the error measure:

$$\xi^2 = \sum_{i=1}^{N} d_i^2 \tag{6.29}$$

Where d_i is the perpendicular distance from the point $\mathbf{p}$ to the line $\mathbf{r}_i$, assuming that the direction vector $\mathbf{b}_i$ is a unit vector then we have:

$$d_i^2 = |\mathbf{p} - \mathbf{a}_i|^2 - ((\mathbf{p} - \mathbf{a}_i) \cdot \mathbf{b}_i)^2 \tag{6.30}$$

Fig. 6.17 provides an explanation of the error measure from a geometric viewpoint. The point $\mathbf{a}_i$ is a general point located on the line, and $\mathbf{b}_i$ is the unit direction vector of the line. The distance d_i^2 is the perpendicular distance between an arbitrary point $\mathbf{p}$ and the line $\mathbf{r}_i$. The origin of the world coordinate system is defined by $\mathbf{O}$. Evaluating the partial derivatives of the summation of all d_i^2 with respect to x, y and z results in the equation for computing the least squares estimate of $\mathbf{p}$:

$$\xi^2 = \sum_{i=1}^{N} d_i^2 = \sum_{i=1}^{N} \{|\mathbf{p} - \mathbf{a}_i|^2 - ((\mathbf{p} - \mathbf{a}_i) \cdot \mathbf{b}_i)^2\} \tag{6.31}$$

Rearrangement of above equation leads to:

$$\frac{\partial \xi^2}{\partial x^2} = \sum_{i=1}^{N} \{2(x - a_{ix}) - 2(\mathbf{p} - \mathbf{a}_i) \cdot \mathbf{b}_i b_{ix}\} \tag{6.32}$$

$$\frac{\partial \xi^2}{\partial y^2} = \sum_{i=1}^{N} \{2(y - a_{iy}) - 2(\mathbf{p} - \mathbf{a}_i) \cdot \mathbf{b}_i b_{iy}\} \tag{6.33}$$

$$\frac{\partial \xi^2}{\partial z^2} = \sum_{i=1}^{N} \{2(z - a_{iz}) - 2(\mathbf{p} - \mathbf{a}_i) \cdot \mathbf{b}_i b_{iz}\} \tag{6.34}$$

$$\frac{\partial \xi^2}{\partial x^2} + \frac{\partial \xi^2}{\partial y^2} + \frac{\partial \xi^2}{\partial z^2} = 0 \tag{6.35}$$

Using matrix notation an equation can be derived to minimise the geometric error function for all N lines.

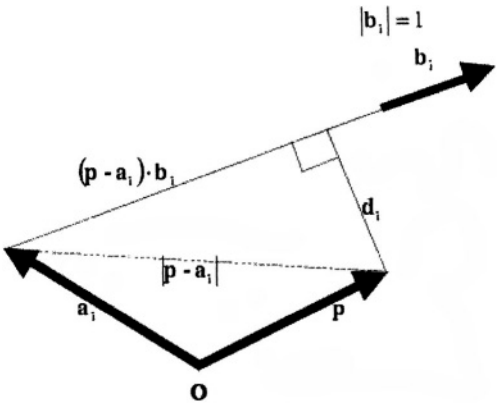

Figure 6.17. Geometric view of the minimum discrepancy.

Acknowledgments

This work was partly undertaken with support from the Engineering and Physical Science Research Council (EPSRC) under grant number GR/M58030.

References

[1] Black J, Ellis T, "Multi Camera Image Tracking", *PETS2001,* Kauai, Hawaii, December 2001.

[2] Black J, Ellis T, "Multi Camera Image Measurement and Correspondence", *Measurement,* 32, 2002, pp. 61-71.

[3] Buxton H, Gong S, "Visual surveillance in a dynamic and uncertain world", *Artificial Intelligence,* 78: pp. 431-459, 1995.

[4] Ellis T, "Co-operative computing for a distributed network of security surveillance cameras", *IEE European Workshop. Distributed Imaging* (Ref. No.1999/109). IEE, London, UK; 1999; 136, pp. 10/1-5.

[5] Ellis T, Xu M, "Object detection and tracking in an open and dynamic world", *PETS2001,* Kauai, Hawaii, December 2001.

[6] Ellis T, "Performance Metrics and Methods for Tracking in Surveillance", *PETS2002,* Copenhagen, pp. 26-31, May 2002.

[7] Ellis T., Makris D. and Black J., "Learning a multi-camera topology", Proc. *IEEE Joint Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance,* 2003.

[8] Hartley R, Zisserman A, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2001.

[9] Heckerman D., "A tutorial on learning with Bayesian Networks", *Technical Report,* MSR-TR-95-06, Microsoft Research, 1995.

[10] Lee, Romano R, Stein G, "Monitoring Activities from Multiple Video Streams: Establishing a Common Coordinate Frame". *IEEE Trans, on PAMI,* Vol. 22, No. 8, August 2000, pp 117-123.

[11] Mikic I, Santini S, Jain R, "Video Integration from Multiple Cameras", *DARPA Image Understanding Workshop,* Monterey, CA November 1998.

[12] Pearl J, Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.

[13] Stauffer C, Grimson W E L, "Adaptive background mixture models for real-time tracking", *Proc. CVPR'99,* 1999, pp. 246-252.

[14] Stein G, "Tracking from Multiple View Points: Self-calibration of Space and Time". *DARPA IU Workshop,* 1998, pp 1037-1042.

[15] Tsai R, "An efficient and Accurate Camera Calibration Technique for 3D Machine Vision", *IEEE Conf. on CVPR,* June 1986, pp 323-344.

[16] Xu M, Ellis T, "Illumination-invariant motion detection using colour mixture models", *Proc. BMVC,* pp. 163-172, Manchester, Sept. 2001.

[17] Xu M, Ellis T, "Partial observation vs. blind tracking through occlusion", *Proc. BMVC,* pp. 777-786, Cardiff, 2002.

[18] Xu M, Ellis T, "Tracking occluded objects using partial observation", *Acta Automatica Sinica,* special issue on Visual Surveillance of Dynamic Scenes, 29(3), pp. 370-380, 2003.