PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 675–689

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Distributed intelligence for multi-camera visual surveillance

P. Remagnino, A.I. Shihab*, G.A. Jones

School of Computing and Information Systems, Kingston University, Penrhyn Road, Kingston upon Thames/KT1 2EE, UK

Received 16 September 2003; accepted 23 September 2003

Abstract

Latest advances in hardware technology and state of the art of computer vision and artificial intelligence research can be employed to develop autonomous and distributed monitoring systems. The paper proposes a multi-agent architecture for the understanding of scene dynamics merging the information streamed by multiple cameras. A typical application would be the monitoring of a secure site, or any visual surveillance application deploying a network of cameras. Modular software (*the agents*) within such architecture controls the different components of the system and incrementally builds a model of the scene by merging the information gathered over extended periods of time. The role of distributed artificial intelligence composed of separate and autonomous modules is justified by the need for scalable designs capable of co-operating to infer an optimal interpretation of the scene. Decentralizing intelligence means creating more robust and reliable sources of interpretation, but also allows easy maintenance and updating of the system. Results are presented to support the choice of a distributed architecture, and to prove that scene interpretation can be incrementally and efficiently built by modular software.

Keywords: Scene understanding; Hidden Markov models; Multi-agent systems; Machine learning

1. Background: visual surveillance and the agents paradigm

Monitoring applications are rapidly increasing thanks to advances in technology and state of the art research. The enormous improvement and affordability of hardware and the availability of distributed computing technologies have encouraged an increasing use of distributed and parallel systems in monitoring applications [1,2]. The popularity of distributed systems is also due to the steady and rapid progress in camera technology (i.e. on-board additional processing or intelligence as it is commonly called) and the availability of networks of cameras covering wider monitoring areas. State of the art research in computer science disciplines, such as artificial intelligence and computer vision has greatly contributed to the success of advanced monitoring systems. Modern visual surveillance systems use networks of cameras increasingly more sophisticated, endowed with additional processing power. Overall the progress is rapidly moving towards a naturally distributed system, where distribution is not only physical, but also semantic. Multi-camera monitoring systems require an intelligent infrastructure, capable of optimal scheduling of visual tasks, performing locally standard image processing, extracting relevant information, and broadcasting or sharing information only on demand (either issued by the user or by other cameras or networks of cameras). A large surveillance system might be scattered over a large physical area and, ideally, must offer different types of service to a large variety of users. For instance, a security guard is interested in identifying potential crime events such as breaking into secure areas, tampering with parked vehicles, or violent and destructive behaviour in public spaces. Shopping mall operators, on the other hand might be interested in determining customer flow to specific shops and through shopping aisles. In addition, though it is outside the scope of this paper, the topic of integrating part of the architecture

* Corresponding author. Tel.: +44-20-8547-7661; fax: +44-20-8547-7824.
E-mail address: a.shihab@kingston.ac.uk (A.I. Shihab).

through intelligent user interfaces is quite important (see Refs. [3,4]).

In *visual surveillance* applications, the user imposes a set of constraints on the installation topology (number, location and orientation of cameras) and the typology of events to identify and monitor over time. Fundamentally, a distributed problem (see [5–7] for more details on distributed intelligence) such as visual surveillance lends itself very well to *artificial intelligence* techniques. This is for a variety of reasons:

- to provide a variable degree of autonomy,
- to adapt to *internal* conditions imposed or generated by the system architecture such as hardware and software failure,
- to adapt to *external* constraints such as changes in lighting and weather conditions.

The need to distribute intelligence—and in particular to move part of the intelligence closer to a camera—may be further justified by the following requirements:

- *Speed*: Lower-level peripheral problem solving not only makes the processing chain parallel but also leaves the central processing nodes to concentrate on more sophisticated *analysis* e.g. *threat assessment*.
- *Bandwidth*: Peripheral processing avoids the need to transmit high-bandwidth and highly redundant data streams to central processing nodes.
- *Redundancy*: Increased reliability since distributed computing platform may be re-configured to handle component failures.
- *Autonomy*: The crucial characteristic of a truly distributed system. A large surveillance system is composed typically of a variety of sensors scattered across a potentially large geographic area. In order to process video data streams asynchronously, using a variety of communication modalities (wireless/wired and analogue/digital), different semantic levels of knowledge are required to be generated by different types of users.

The central development that has enabled the design of truly autonomous distributed artificial intelligent systems is the *agent* paradigm [8–12]. The *agent* paradigm extends the *object-oriented* paradigm [13,14]. An object, the basic passive element commonly used in object-oriented design is augmented to create *agents*—self-contained software modules capable of independent reasoning and self-adaptation to an evolving external world. Despite the often-heated nature of the debate over the definition of agents, there are some agreed characteristics. Agents must be autonomous and capable of exchanging knowledge about the environment. An agent must be able to adapt its plans to both internal changes (e.g. faults) and external (environmental) changes, by modifying its own beliefs (model) of the perceived world, and adapt its own desires (goals).

The use of the agent paradigm in computer vision has increased in the last decade or so [15,16]. Computer vision requires *intelligent* software to extract visual data and encode it in the simplest form. Scene understanding entails the extraction of semantic information about the viewed scene. The process of bridging the gap between data and information, to go from low-level image analysis to high-level interpretation, is the biggest challenge in vision research.

The agent paradigm lends itself very well to a distributed intelligence system capable of an automatic interpretation of a dynamic scene based on computer vision techniques. The work of Buxton [17] is a good example of an event-based detection system for surveillance situations using Bayesian networks as a simple yet effective classification technique. Intille [15] used a more orthodox agent-based system to classify American football actions again using Bayesian networks to identify the events. Remagnino [16] used Bayesian networks to model behaviours and interactions between objects in a typical car park scenario. Rosario [18] has modeled interactions using coupled hidden Markov models that assume that interaction behaviours are well represented by interwoven stochastic processes.

Due to the uncertain, partial and noisy nature of the imaging process, the agent architecture proposed by this paper make extensive use of probability theory. Probability theory is the common language used by the agents to store information about the environment, to reason about the environment, and communicate knowledge about the environment to other agents. Researchers in artificial intelligence have been increasingly using probabilistic methods to deal with the uncertain and incomplete nature of real-world applications making particular use of graphical models such as Bayesian networks and hidden Markov models [15–17,19,20]. These methods lend themselves very well to the implementation of distributed intelligent systems. They can be used to represent and reason about knowledge following formal probabilistic calculus; they can be used to implement robust and complex communication protocols and are a natural way of inferring new information from old information and newly acquired data. A number of researchers have used probabilistic techniques such as Bayesian networks [21,22] and hidden Markov models [23,24] to infer natural language type descriptions of events in traffic scenarios [16,17], tennis [25] or football matches [15], office dynamics [20], sign language modelling [19] and people behaviours [18,26].

In Section 2 we provide an overview of our proposed agents architecture. In Section 3 we provide detailed descriptions of individual agents. Sections 4 and 5 illustrate a partial implementation of the architecture, illustrating how a statistical model incrementally built by the multi-agent society can capture scene dynamics. Section 6 summarises the paper.

2. Proposed agents architecture for visual surveillance

At the outset, let us clarify that in this paper, we lay out the specification for our agent architecture and expand on some aspects of its implementation. We present here a preliminary treatment of how to develop an agent community for visual surveillance system.

The installation of a typical monitoring system is designed around a secured environment monitored by a number of cameras with overlapping views (see pictorial representation of Fig. 1). A number of PCs distributed around the site are expected to process the incoming video. A *camera agent* responsible for detecting and tracking all moving objects traversing the camera's view volume manages each camera. For each new event (initiated by a scene object entering a camera's view volume), an *object agent* is instantiated with the responsibility of providing a continuously updated description of the temporal event. All cameras are calibrated to a common ground plane coordinate system to enable information to be integrated across multiple cameras (see later discussion on how camera agents co-operate to work out a joint calibration).

For each temporal event, the *camera agent* computes the 3D trajectory of the object and extracts the set of sub-images containing the event in each frame. These data are continuously streamed to the appropriate *object agents* and used to update a classification of the event object itself—e.g. vehicle person—and a semantic commentary on the object behaviour. Since an event may be detected in more than one field of view, multiple *object agents* may be instantiated for a single event. Consequently *object agents* must also collaborate to identify these multiple instantiations. Any pair of *object agents* that detect such equivalence merges to create a single *object agent*, inheriting data channels from multiple *camera agents*. *Object agents* terminate once tracking information from attached *camera agent*(*s*) has been terminated (see Ref. [27] for a more detailed discussion).

Each independent agent is effectively implemented as a separate process, processing data streamed by a camera, extracting only the currently relevant information (mainly regions and objects of interest) and then sharing such information across a cluster or more clusters, or the entire network of sensors. Camera agents having overlapping views interact to merge extracted information, effectively creating separate societies of agents, with their own agendas, determined by the immediate request of the user, or a basic set of tasks imposed by the system administrator. A typical agenda for a monitoring system will include items such as: detecting unusual dynamics, keeping up-to-date the current normality of movements in the scene, and logging events parameterised to the day of the week and time of the day. The last item is justified by the known existence of patterns of activity that occur at almost regular intervals in common monitored areas, such as public spaces.

2.1. Internal architecture of agents

Visual surveillance agents do not have intentions or desires, however they work on uncertain beliefs about the *perceived* world and they co-operate to achieve an optimal interpretation of the scene. The goal is therefore unique, even though it can be segmented into separate sub-goals, representing the interpretation of areas of the scene. Competition also: object hypotheses are generated and interpretation implies conflict resolution about two or more contradicting hypotheses (based on positional and velocity information), the resolution of which is either imposed by a central agent, such as the ground plane agent, or more slowly by a peer-to-peer communication. Each agent has their own view or belief of the scene: so for instance, the camera agent triggers image processing algorithms to learn a model of the background and then detect motion in its field of view. Each motion event in turn either creates an object agent, or establishes a communication with object agents already existing. Over time the camera will learn its position and orientation with respect to the ground plane. Again, this is either implemented as a direct communication between camera and objects' agents, or developed with the intermediary ground plane agent. As mentioned earlier on, the real purpose of the ground plane agent is to cut down the computational load.

The agents we designed are tabulated in Table 1. Each agent has their own beliefs and goals; all agents interact to achieve the same global goal of interpretation of the scene. Beliefs are based on the uncertain information generated by the image processing algorithms. Agents are aware of other agents only if this is necessary for the overall goal of the community. The ground plane agent is aware of all the camera agents and all object hypotheses; object hypotheses are asked to communicate when the ground plane agent decides to garbage collect an area of interest, or new data arrives to suggest so. Cameras and object hypotheses are communicating to stream newly gathered information about the scene.

2.2. Agent communication mechanisms

We did not adopt a specific agent communication language at this stage of our work. As stated in the beginning of this section, the communication mechanisms we suggest are preliminary. We classify communications between agents as either "relevant" or "irrelevant". Relevant communications are those that concern the same scene; for example, position and velocity measurements between camera and object agents working on the same scene. Irrelevant communications are identified on the basis of the results of the visual registration process: data streams pertaining to different scenes score low registration results. Therefore, agents that receive irrelevant communications, discard them. A disadvantage of this process is the computational overheads of classifying the exchanges between the agents.

Our agents currently integrate information on the basis of Bayesian calculus as detailed in the next section.

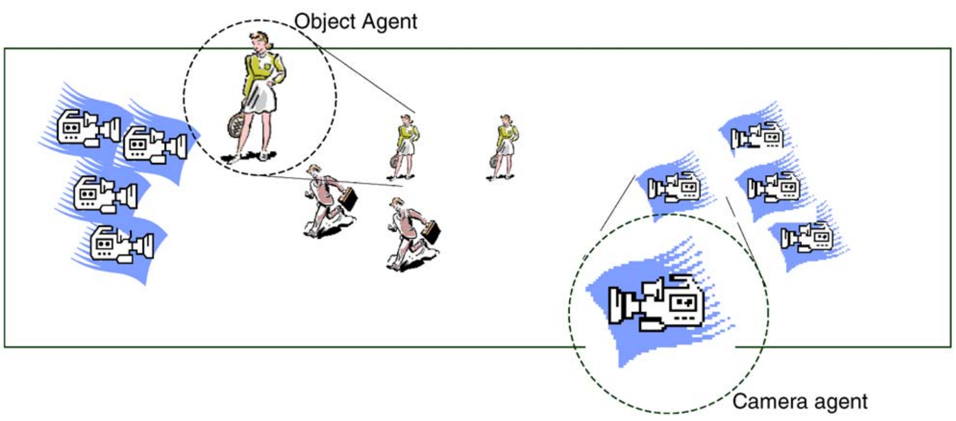

Fig. 1. Multi-camera monitoring scene.

Table 1
Agents used in our system

Agent	Goals	Beliefs
Camera	Self-calibration, registration, event detection	Pose *wrt* ground plane, events, frequency map (hot spots)
Object hypothesis	Updating bearings	Position and velocity
Ground plane	Mediate calibration, registration, and hypothesis consistency maintenance	Cameras poses and scene interpretation

Communications are streamed directly or indirectly (through the ground plane agent) between two agents relevant to the same measurement. Uninterested agents reject proposed communication, while interested agents may accept the communication but they might decide not to integrate newly streamed data, because it is too far from the current hypothesis, in accordance with the Bayesian rules.

2.3. Ontology

Work on a suitable ontology of communication must also take into account the impact and the use of the system. Different categories of users will adopt a different way of describing an occurring event, and the authors believe the ontology shared by visual surveillance agents ought to be strongly user-centric. The first author has already worked on the automatic interpretation and delivery of surveillance data [28,29]. The idea is to identify a library of terms that can be employed to infer interpretation of the dynamics of a scene, and, at the same time, provide a dictionary sufficiently rich to inform a user of the evolution of a scene. The authors have chosen a geometric ontology as the basis of discourse: all agents, including the user interface agent—that will have to mediate with the user—understand basic geometric concepts. Examples include a rough position and velocity of objects seen in the scene. (For instance passing in the field of view of camera X, with high speed, crossing its field of view from left to right, or two events crossing in the field of view of camera X and Y and standing still for a while).

3. Description of camera and object agents

We now describe the components and functions of the camera and object agents.

3.1. Camera agent

Currently our camera agents are hosted on PCs using the camera as a resource. As intelligent cameras become computationally more powerful, the centre of gravity of a camera agent may well migrate towards the camera. In addition to the instantiation of *object agent* processes and the reliable transport of trajectory and pixel information to these objects, to track events reliably, each *camera agent* must also enjoy sufficient algorithmic intelligence to perform the following tasks: calibration, tracking, and learning.

Camera calibration: The mapping between an image plane and the ground plane (the homography) [30] must be continually maintained, especially when the sensor infrastructure also comprises multiple pan, tilt and zoom (PTZ) cameras. Security operators wishing to use the PTZ cameras to investigate interesting or suspicious events periodically interrupt the described agent system. The calibration procedure is repeated automatically when a camera returns to its *home* position to cope with positional inaccuracies associated with typical pan, tilt and zoom mechanisms.

The procedure to recover the full image to ground-plane homography is time-consuming, labour-intensive and skill-dependent. Rather than rely on such an onerous procedure, we propose instead to use a simple learning procedure that observes examples of the typical object types within a surveillance scene and then recovers the relationship between image position and the projected width, height and motion of an object. This learning calibration procedure utilises the simple but reasonably accurate assumption that in typical surveillance installations—*the projected 2D image height of an object varies linearly with its vertical position in the image*—from zero at the horizon to a maximum at the bottom-most row of the image. This height model is derived from the optical geometry of a typical visual surveillance described in Ref. [31]. In addition, such an assumption enables the use of simple but highly discriminatory models of the appearance of scene objects, which indirectly use the depth of the object to model its projected width and height. Since the spatial extents of objects are now a function of image position, any image tracker will be more robust when presented with the distorted observations, which arise from fragmentation or occlusion processes. Our tracking model represents an object simply as projected 2D rectangles whose height and width are determined by the object's 3D position in the scene and hence 2D position in the image [31].

Tracking: The *camera agent* is charged with the detection and tracking of all moving objects within the field of view. First the pixels in each new frame are classified as *moving* or *not-moving* by comparison with a temporally updated mean and variance frame [32]. Regions within the resultant *event map* are extracted and used to validate current events. Regions, which cannot be accounted for by current events, are used to hypothesise a new event using an expected appearance model [33–35]. Event hypotheses, which are continuously validated for a preset number of frames, are promoted by instantiating a new *object agent*. While introducing latency in the generation of scene knowledge, such a threshold is necessary to avoid the costly overhead of creating new *object agent* threads/processes for false alarm events. In fact, the value of this threshold is somewhat critical. Too low a value may result in spurious objects being instantiated in response to short burst of spatially correlated noise. Large values prevent the object from benefiting from any available data fusion from overlapping views.

Learning: This entails the learning and updating of statistical models of object appearance that change with external lighting and weather conditions. Typical parameters that can be learned include the height, width and shadow dimensions used in the tracker. An image-plane to ground-plane geometric relation can be estimated, assuming some knowledge about the scene, for example, the approximate height of the camera and height of pedestrians walking on the ground plane. The estimation works on the assumption that one can identify pedestrians and track them throughout the scene. A learning technique can then be applied to calculate incremental estimates of the parameters required to establish the image plane to ground plane transformation [31]. After some simple algebraic manipulations the ground plane position (X, Y) of an object can be derived as follows:

$$X = \frac{\alpha_x^f (j - j_0) L}{\alpha_y^f (i - i_0) \sin\theta - \cos\theta},$$

$$Y = \frac{-L(\alpha_y^f (i - i_0) \cos\theta - \sin\theta)}{\alpha_y^f (i - i_0) \sin\theta - \cos\theta}.$$

Thus to compute the real-world ground plane position of an image point, the intrinsic and extrinsic camera parameters $i_0, j_0, \alpha_x^f, \alpha_y^f$ (the optical centre co-ordinates and the horizontal and vertical inter-pixel widths, L being the estimated height of the camera from the ground plane) and θ (the tilt angle of the camera) must be estimated. In our approach, the optical centre i_0, j_0 is computed by an optical flow algorithm that robustly fits a global zoom motion model to a three-frame sequence undergoing small zoom changes. The rest of the parameters may be recovered in the following training procedure based on watching events unfolding within the monitored scene.

For the zero-roll optical configuration, the projected image height of an object may be captured as a linear model varying against vertical image position only. In pixel coordinates, this linear relationship may be expressed as

$$\mu = \gamma(i - i_h),$$

where μ is the object height in pixels, γ is the *height expansion rate* (HER), and $\mu = 0$ at the horizon i_h (see [31] for more details). This model may be extracted from the scene automatically by accumulating a histogram $H[i_n \mu_n]$ from a large number n of object observations of the monitored scene. A standard motion detection process is employed to extract connected components of moving pixels [32].

3.2. Object agent

Once instantiated, the *object agent* is obliged to pursue the goal of classifying the event with which it is associated and classifying its behaviour from a limited number of domain-specific activities. This knowledge is derived from the trajectory and pixel information provided by the *camera agents*(s). A trajectory estimator is employed to integrate the stream of 2D ground plane observations—possibly supplied by more than one *camera agent*—to generate smoothed position and velocity estimates of object. In addition, the

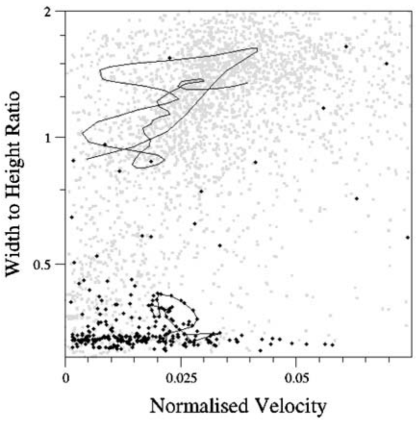

Fig. 2. Width to height ratio versus normalized velocity.

projected 3D width and height of an object can be temporally tracked by combining the region dimensions extracted from the event map with the position of an object on the ground plane. Classification and behavioural analysis is performed by the following intelligent algorithms:

Object classification: People and vehicles enjoy distinct 3D velocity and 3D projected width characteristics. Fig. 2 illustrates the class conditional probability distributions of vehicle and people in this *classification* space. Since to some extent these distributions are overlapping, it is necessary to integrate velocity and width observations over the history of a tracked object to avoid false classification. This can be illustrated by overlaying the *trajectories* in *classification* space of a typical person and vehicle event. A simple maximum a posteriori decision rule is employed to update the probability of classification given each new observation $a_t = (v_t, w_t)$ of the velocity v_t and width to height ratio w_t.

$$\omega^* = \operatorname*{argmax}_{\omega \in \Omega} P(\omega | a_t, \dots, a_1), \tag{1}$$

where Ω is the set of possible classifications $\Omega = \{person, \ vehicle\}$. Assuming each observation a_t is independent of previous observations, the a posteriori probability $P(\omega|a_t, a_{t-1}, \dots, a_1)$ may be expressed temporally as

$$P(\omega|a_t, \dots, a_1) \propto p(a_t|\omega) P(\omega|a_{t-1}, \dots, a_1)$$

$$P(\omega|a_1) \propto p(a_1|\omega) P(\omega). \tag{2}$$

The class conditional probabilities $p(a_t|person)$ and $p(a_t|vehicle)$, and prior probabilities $P(person)$ and $P(vehicle)$ are derived from training data such as that shown in Fig. 2.

Comparison of object agents: Identifying *equivalent objects* in multiple cameras can be handled within the present agent framework by first restricting the responsibility of the camera agents to invoking object agents for each new event in the scene, and second giving responsibility for identifying equivalent events to the object agents. All object agents are required to communicate with each other and compare trajectory information. Two object agents, which have highly similar trajectories, negotiate to create a new merged object agent before terminating themselves. The new agent from each of the merged agents inherits the original data streams from each camera. Such a new agent will continue as before to update its trajectory from this combined source. Comparing the 3D ground plane positions over time can identify identical trajectories. Given two trajectories $\boldsymbol{X}$ and $\boldsymbol{X}'$, the positional separation $\delta \boldsymbol{X}$ is given by $\|\boldsymbol{X} - \boldsymbol{X}'\|$. Objects whose trajectories are separated by a distance less than some threshold τ are considered identical. In Fig. 3(a), the separation $\delta \boldsymbol{X}$ between two object trajectories is plotted over time for a single event seen from two different cameras. In comparison, Fig. 3(b) shows the separation between two non-corresponding trajectories. Generally, the separation distance between matching trajectories is much lower than non-matching trajectories, although the occasional noisy trajectory estimate generates low separation even between non-matching trajectories. Thus the merging of agents is only initiated after the trajectories have been repeatedly determined as identical for T seconds (where T is currently set at 5 s). Introducing such an inertial characteristic ensures that the costly process of merging agents is not triggered by occasional gross noise conditions. (In practice the control structure for dealing with more than two overlapping views is non-trivial. Currently, only pairs of object agents may merge and reset (i.e. forget) any other negotiation involving the merged pair. It is intended to explore the possibility of delaying merge operations if more than two object agents flag a potential merge.)

Another way of merging information is by learning the correspondence between two cameras with partially overlapping fields of view. This can be implemented by having two camera agents exchange information on their extrinsic parameters, that is how their image plane relates with the ground plane, and then using a shared voting mechanism to estimate a single Cartesian frame on the ground plane. Details of this technique are discussed in Ref. [31]. Briefly, the Hough Transform approach [36] can be adopted to recover the inter-camera transformation by taking advantage of the fact that the ground plane coordinate systems of temporally synchronised observations of the same 3D object are related by a simple rotation ψ and translation T transformation.

$$X' = R(\psi)X + T,$$

$$V' = R(\psi)V,$$

where X, V and X', V' are positional and velocity observations measured in the local ground plane co-ordinate system (GPCS) of two cameras C and C', respectively. In every frame interval, each camera outputs a set of measurements about all objects located in its field of view. As object correspondences are unknown, every pair of observations from each of the cameras agents can be exchanged to generate a candidate estimate of the transformation. Given a pair

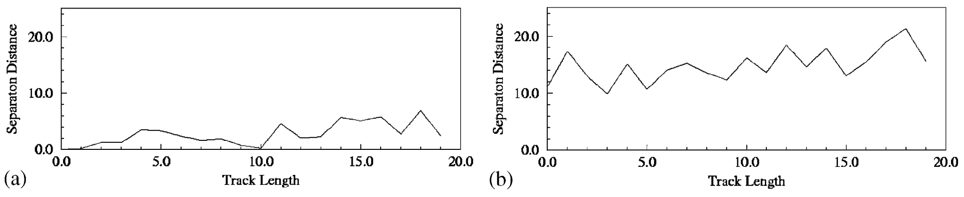

Fig. 3. Separation distances over 20 frames (i.e. approx. 1 s) for (a) two trajectories representing the same event seen in two cameras, and (b) two different events.

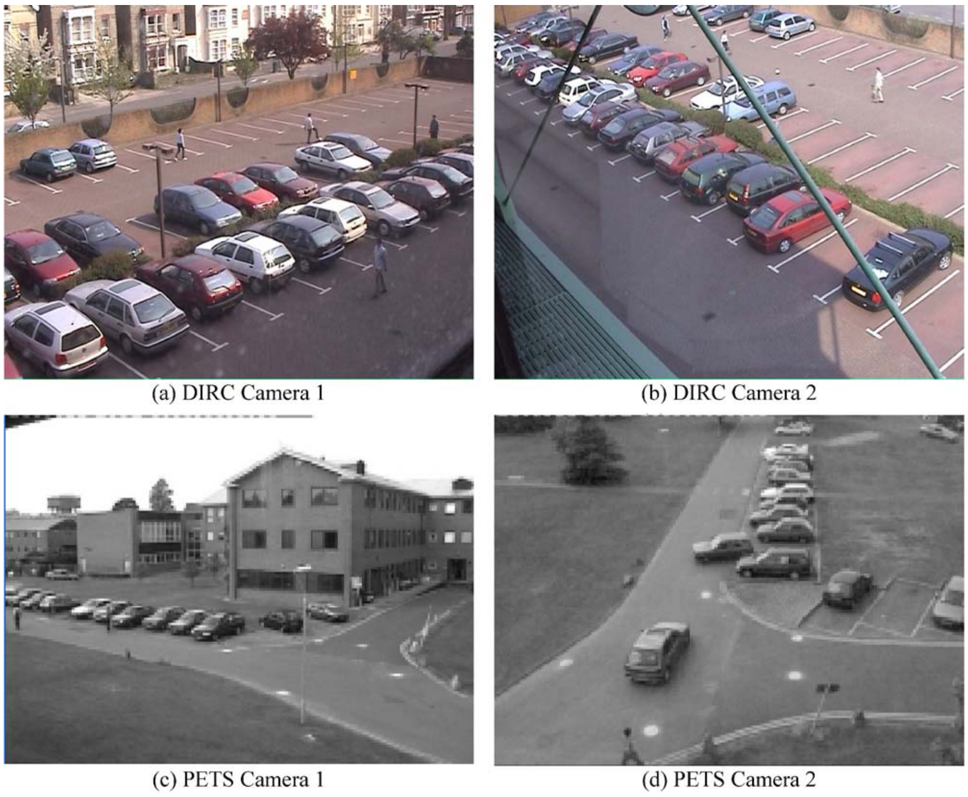

Fig. 4. Two test data sets: DIRC (a) and (b), and PETS (c) and (d).

observations $X'_{t,i}$, $V'_{t,i}$ and $X_{t,j}, V_{t,j}$ at time t from camera C' and C respectively, transformation estimates may be defined as

$$\cos\psi_{t,i,j} = \hat{V}'_{t,i}\hat{V}'_{t,j},$$

$$T_{t,i,j} = X'_{t,i} - R(\psi_{t,i,j})X_{t,j},$$

where $\hat{V}$ is the unit vector along the direction V. If Λ_t and Λ'_t are the set of observations in frame t for cameras C and C' respectively, then the set of all observations

$$\{\psi_{t,i,j}, T_{\tau,i,j};\ \forall i \in \Lambda_\tau,\ \forall \tau \leqslant t\}$$

should ideally exhibit a distinct cluster of estimates around the true solution within the noise floor of uncorrelated false estimates generated by incorrectly corresponded observation pairs and noise observations. To detect this cluster, the space could be tessellated into bins and a Hough transform technique applied to locate the maximum that correspond to the optimal transformation parameters. However, the range of translation values required is difficult to predict a priori. Therefore to avoid the storage problems such a voting strategy introduces, a robust clustering approach is adopted. The expectation–maximisation mixture of Gaussian technique was implemented and adapted to iteratively perform the cluster analysis on the incoming stream of transform estimates. The cluster continually reports the most likely transformations between cameras (Fig. 4).

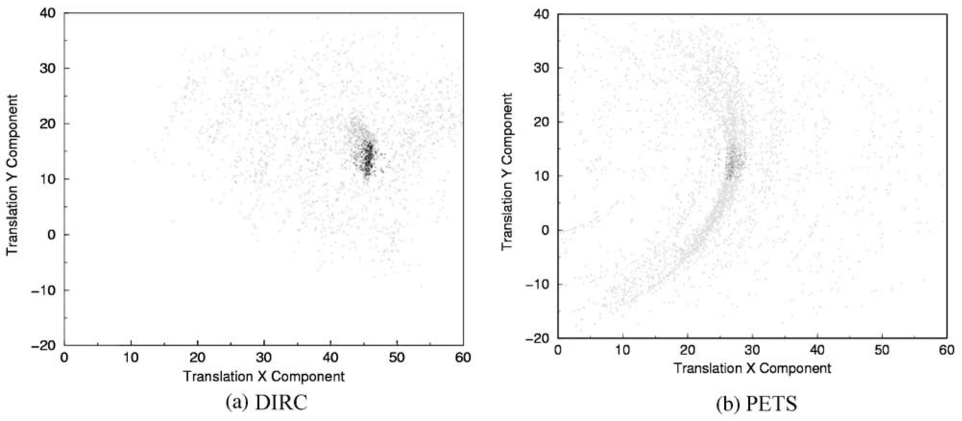

Fig. 5. Locating the maximal cluster in two example Hough space (Translation) for two test data sets (a) DIRC Data sets (b) PETS Data sets.

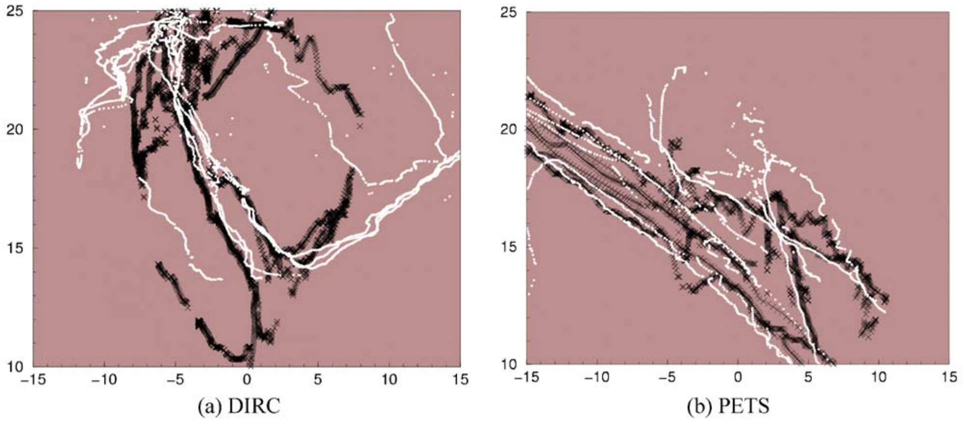

Fig. 6. Aligned ground plane trajectories.

Examples of the populated Hough space and dominant cluster are shown in Fig. 5(a) and (b) for the DIRC and PETS data sets (see Fig. 4), respectively. Note that these peaks are robustly recovered from an extensive noise floor. The equivalent overlapped trajectories from each view are shown in Fig. 6. The black and white 3D positions are recovered from two very different viewpoints.

A more quantitative evaluation of the accuracy of the technique may be judged by comparing the registration results of the method with the equivalent data generated by the Tsai calibration method (and the survey ground truth for the DIRC data sets). Table 2 plots the rotation angle ψ (in degrees) and distance $\|T\|$ (in metres) between the origins of the two local GPCS for the TEST and PETS data sets. Results require more extensive validation—particularly as the Tsai results themselves are extremely sensitive to the number and distribution of reference points within the image and view volume. Nonetheless the agreement is promising given the fact that no manual correspondence or measurement is required in the proposed *learning* method.

Table 2
Comparison of proposed method and Tsai method

Data sets	Measurements			
	Tsai calibration [41]		Proposed Hough technique	
	ψ	$\|T\|$	ψ	$\|T\|$
DIRC	75	57.2	81	53.5
PETS	76	27.9	70	29.5

Behavioural Classification: *Object agents* are also responsible for identifying the behaviour of the event with which it is associated. The approach adopted here is to employ the trajectory information of an object to identify the likely behaviour of the event for an application specific domain such as *car entering*, *person leaving car*, *car parking*, etc. The hidden Markov model (HMM) methodology [23] is used to construct stochastic models of behaviour during a training phase. At run-time, a newly created *object agent* uploads the relevant models (i.e. some models are more probable at different time of the day) and repeatedly computes the likelihood of each model given the current trajectory. Such an approach not only provides a complement event classification technique but also, crucially, identifies the event behaviour. The HMM approach is described more fully and evaluated in the next section.

4. Scene modelling using hidden Markov models

In this paper scene events are sequences of sub-images of objects moving in the field of view of cameras. Event detection of a moving object in the scene triggers a number of hypotheses related to its presence, shape and position. All such hypotheses are then assigned to agents. In the proposed framework *object agents* continuously co-operate to provide an optimal interpretation of the scene. The scene model is learned using hidden Markov models (HMMs). Models are learned for different categories of objects in a training phase, however such models are automatically updated by the observations of new tracked events. All tracked events contribute to the overall interpretation of the analysed scene, updating the most probable activity model.

The underlying idea of building a scene model has a number of advantages. First and foremost the incremental construction of the scene model from raw data. Such bottom-up interpretation can then be used to guide object classification. Learned models can be used to determine what object type originated the event, by calculating the likelihood of each model conditioned to the current set of observations. More importantly, the use of a scene model allows the agent to predict all feasible object future trajectories, sort them by likelihood and, potentially, feed back useful information to the tracking process. At this moment in time co-operation is limited to an exchange of geometric and chromatic features, but even in this form communication defines a rigorous object ontology with a formal vocabulary and pre-defined methods for integrating information.

In addition to maintaining an estimate of its trajectory, each *object agent* attempts to interpret the activity of its event from a number of predefined activities that have been identified for the particular problem context. For example, in our car park application, the set of allowable activities include

- *People behaviour*: walking through or near the car park, reaching or getting out of a vehicle.
- *Vehicle dynamics*: passing near, through or parking in the car park.

The proposed approach makes use of the Markov learning technique. A pre-classified training set of tracks for people and vehicles is acquired from which Markov models of the required activities occurring in the analysed environment are built. At run-time, each event agent loads a copy of the scene model (all built HMMs). For each set of observations, corresponding to the object tracks, the corresponding event agent calculates the likelihood of all the available models (and hence activity) to identify the most likely activity of the object, and its class.

4.1. Markov models for scene understanding

The Markov model is a probabilistic technique for learning and matching activity patterns. Each Markov model is a stochastic state automaton in which each state is associated with both a prior probability of a process starting in that state, and a set of transition probabilities describing the likelihood of a process moving into a new state. In a previous implementation of our work [37], these states mapped directly onto manually identified regions of interest in the scene by an operator in an offline procedure from a series of semantic labels, e.g. *parking areas*, *vehicle entrance*, *exit*, *pedestrian path*, etc. This process of manual pre-selection is not ideal, as the resultant regions may not necessarily model the probability density of activity accurately. Consequently, the current implementation generates the states automatically by employing the *expectation–maximisation* (EM) algorithm [38] to cluster the activity landscape. The EM algorithm *fits* a number of Gaussian probability distributions (the clusters) to the set of object positions derived from the set of event trajectories in the training set.

Each type of activity for people or vehicle events may be characterised by a family of trajectories moving on the ground plane and, hence, can be represented in a hidden Markov model as a set of states and associated prior and transitional probabilities. During the training phase, the dynamics of objects moving in the scene are used to train each hidden Markov model by computing the following:

- the prior probability π_i for each state S_i, representing the probability that a particular region *state* is the starting point for a trajectory,
- the transitional probabilities a_{ij} between two states S_i and S_j, capturing the probability that an object moves from one state to another given all possible transitions from that region, and
- the probability distribution function $b_j(\mathrm{O})$ of an observation vector O for a state, S_j, i.e. the conditional probability of a particular position O given the state S_j.

In the current implementation, the hidden Markov models are built in two stages. First, for each type of event, e.g. pedestrian or vehicle, the set of states are extracted from the appropriate training set by locating clusters of observations using an efficient implementation of the EM algorithm [39]. Second, trajectories from the training set are used to compute the HMM probabilities. Priors are readily learnt by counting the frequency of starting in a particular state. Similarly transitional probabilities are learnt by counting the number of times a transition between states (including within a state) occurs normalised by all occurring transitions. The formulae used to build the HMMs are normalised to cater for the common problem of probability values rapidly converging to zero. Alternative single step algorithms to build HMMs exist. However, for efficiency, a de-coupled process was adopted.

4.2. Behavioural classification

Once the hidden Markov models have been learned they can be used to describe the dynamic evolution of the scene. Model selection can be performed by finding the model λ which yields the highest a posteriori likelihood $P(\lambda|\mathrm{O})$ given the sequence of N observations $\mathrm{O}=(\mathrm{O}_1,\ldots,\mathrm{O}_N)$ associated with each new trajectory. Reproducing the work of Rabiner [23], this probability can be calculated by introducing the random variable q, a possible sequence of states followed by the trajectory O, and summing over all possible state sequences as follows:

$$P(\mathrm{O}|\lambda)=\sum_{\forall q}[P(\mathrm{O}|\mathrm{q},\lambda)P(\mathrm{q}|\lambda)],$$

where

$$P(\mathbf{q}|\lambda)=\pi_{q_1}a_{q_1q_2}\cdots a_{q_{T-1}q_T}, \tag{3}$$

and $q_1,\ldots,q_T$ represent the temporally ordered sequence of T states in q. This probability of a sequence q given the model λ is the product the probabilities of all state transitions and the prior of starting in the state S_{q_1}. The probability of the observation set O given both the sequence q and model λ is the product of probabilities of the observations $\mathrm{O}_1,\ldots,\mathrm{O}_T$.

$$\mathrm{P}(\mathrm{O}|\mathbf{q},\lambda)=b_{q_1}(\mathrm{O}_1)\cdots b_{q_T}(\mathrm{O}_T)$$

Combining the two equations yields the following expression for the probability of the observation set O given the model

$$P(\mathrm{O}|\lambda)=\sum_{q_1,\ldots,q_T}\pi_{q_1}b_{q_1}(\mathrm{O}_1)a_{q_1q_2}\cdots a_{q_{T-1}q_T}b_{q_T}(\mathrm{O}_T). \tag{4}$$

At each moment in time, each object agent calculates the likelihood of a model given the current set of observed features about the object. The model λ that yields the highest a posteriori probability is the one currently providing the most likely interpretation of object activity, i.e. $\lambda'=\mathrm{argmax}_{\lambda\in\Lambda}P(\mathrm{O}|\lambda)$.

The most likely model is calculated using the classical *forward iterative procedure* provided by the HMM framework [23]. The procedure makes use of intermediate *forward* variables $\alpha_t(i)$, $\forall i$ defined as the likelihood that a set of observations finishing in state S_i are described by the model λ. The procedure proceeds in three stages. In the first *initialisation* stage, $\alpha_1(i)$ is assigned an initial value at the first time interval for all the defined states S_i, $i\in[1,N]$

$$\alpha_1(i)=\pi_i b_i(\mathrm{O}_1),$$

defining the probability of the first observation O_1 given the model and the a priori probability of the state S_i. The second *induction* stage calculates the probability of being in a new state S_j given the previous history of the trajectory for all α variables one step ahead in the future

$$\alpha_{t+1}(j)=b_j(\mathrm{O}_{t+1})\left[\sum_{\forall i}\alpha_t(i)a_{ij}\right]. \tag{5}$$

The process is repeated until the termination stage in which the a posteriori probability of a model λ is computed by summing over all final values of the α variable computed for the model λ

$$p(\lambda|\mathrm{O})\ \propto\ p(\mathrm{O}|\lambda)P(\lambda),$$

$$p(\mathrm{O}|\lambda)=\sum_{\forall i}\alpha_T(i),$$

where $P(\lambda)$ is the a priori probability of the model λ. However, rather than using $P(\lambda)$, the more accurate classification probability $P(\omega|a_T,\ldots,a_1)$ (derived in Eq. (1)) enables the classification procedure to directly influence the selection of the appropriate behavioural model as follows:

$$p(\lambda|\mathrm{O})\ \propto\ p(\mathrm{O}|\lambda)p(\lambda|\omega)P(\omega|a_T,\ldots,a_1), \tag{6}$$

where $p(\lambda|\omega)$ is the conditional probability of a particular behaviour λ given the classification ω of the event.

Two notes remain to be addressed. First, on left–right HMM: if the observed data includes speed information, then the left–right model is definitely the most appropriate Markov model. In our specific examples, the data used to learn the HMM are temporal streams made of only consecutive position measurements. In such a case, the left–right model does not apply. Second, concerning regions versus states: image positional information of a large set of training data is used to generate the probability density function (pdf), representing effectively the frequency map of the scene. The pdf is approximated by the EM/Baum–Welch algorithm, which fits Gaussian components to the modes of the distribution. Each mode is considered as a separate state. However, one can easily group more states having similar semantics into regions. For instance a special example of semantics is spatial semantics, a region might define an entrance of a parking space or a parking lot. In general one can imagine more than one mode, that is more than one Gaussian component, modelling a region in space.

5. Results

5.1. Model building

Some preliminary behaviour classification results are presented. Trajectories for two behaviours—*vehicle entering* and *person entering*—were extracted by hand from over 5 h of video data and used to construct HMM models. Three different sets of models were constructed using the EM algorithm using increasing numbers of Gaussians (i.e. 5, 10 and 20, respectively) to assess the performance for increasingly complex models. Superimposed on a frame from the video sequence, the resultant states are shown as projected one standard deviation ellipses in Figs. 7–12 for each of these complexities on Person and Vehicle models. Specifically, Figs. 7, 9 and 11 present the results for the person model using 5, 10 and 20 Gaussians. Similarly Figs. 8, 10, and 12 present the models for 5, 10, and 20 states, respectively for the vehicle model. Note that the footfalls for the person behaviour model has correctly been identified on the pavement on the left despite the occlusion problems created by the wall. This may be because of the top half of the person (the bounding box) is visible above the wall. It should be noted that, particularly for the *person-entering* model, the object tends to spend more time within a state than moving to a new state.

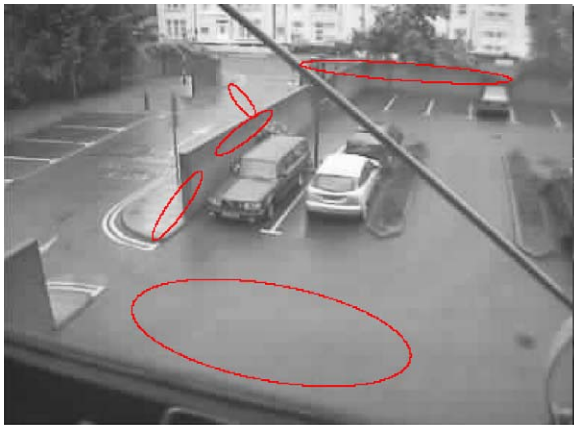
Fig. 7. Person model (5 Gaussians).

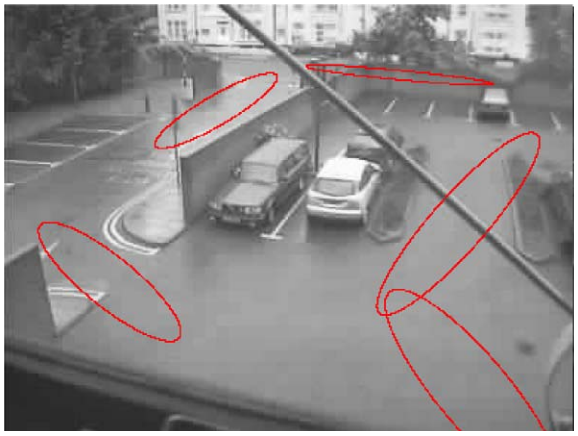
Fig. 8. Vehicle model (5 Gaussians).

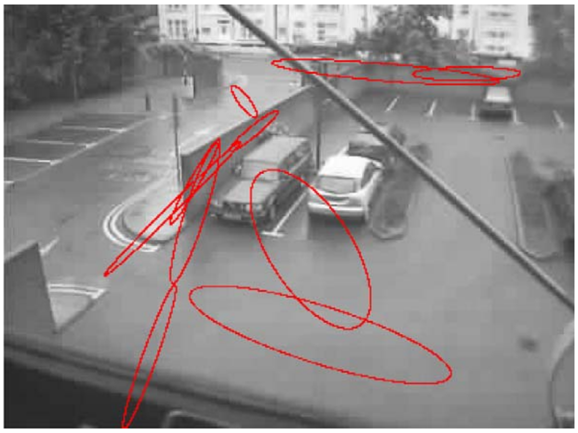
Fig. 9. Person model (10 Gaussians).

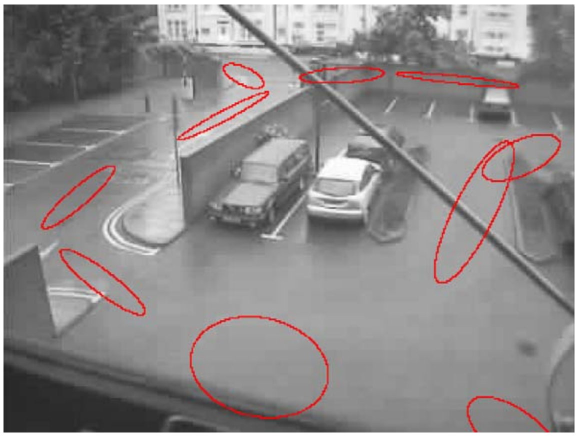
Fig. 10. Vehicle model (10 Gaussians).

5.2. Behaviour classification

To illustrate the effectiveness of the classification process, the models were tested against a vehicle and person set of test trajectories (collected separately from the training data) for each of the three sets of models—5, 10 and 20 Gaussians. Performance may be assessed for each of the three cases by producing a *scatter matrix* detailing the percentage of behaviours correctly and incorrectly explained by the behavioural models. These matrices are presented in Tables 3–5 for the 5, 10 and 20 Gaussian cases, respectively.

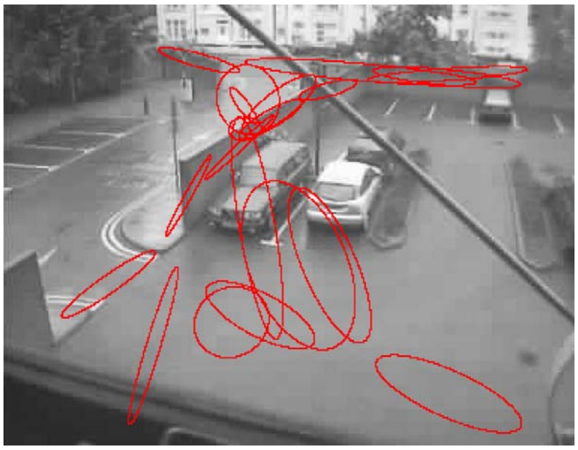

Fig. 11. Person model (20 Gaussians).

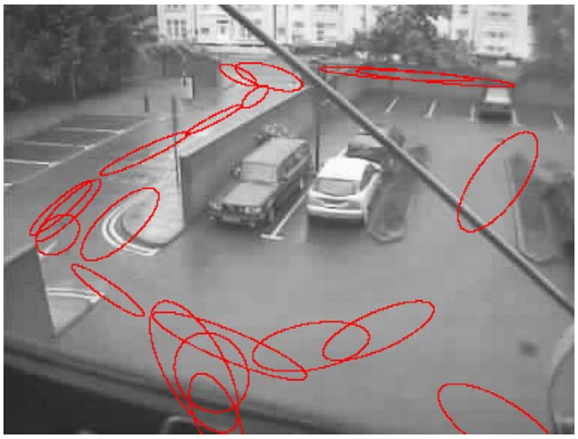

Fig. 12. Vehicle model (20 Gaussians).

Table 3
Classification results for 5 Gaussians

Scatter matrix	5 Gaussians	
	Vehicle model (%)	Person model (%)
Vehicle data	52	48
People data	39	61

Inspection of these tables indicates that as the number of Gaussians used to model the activity on the ground increases, the classification accuracy rises. For the 5-state model, the EM algorithm has poorly modelled the activity within the scene resulting in the essentially random classification of vehicle data.

Classification accuracy can be significantly improved by firstly increasing the number of states. Greater separation is achieved by including greater numbers of states. Such gains are likely to reduce as the number increases beyond 20 states. Indeed there is an increasing likelihood of over-training in which the HMM no longer generalizes but rather begins to model the specific training set. A second improvement can be achieved by adopting the a posteriori probability expression in Eq. (6) that integrates the object classification evidence. Since we have only one activity per object type, this rule may be written as

Table 4
Classification results for 10 Gaussians

Scatter matrix	10 Gaussians	
	Vehicle model (%)	Person model (%)
Vehicle data	63	37
People data	28	72

Table 5
Classification results for 20 Gaussians

Scatter matrix	20 Gaussians	
	Vehicle model (%)	Person model (%)
Vehicle data	77	23
People data	27	73

Table 6
Combined classification results

Scatter matrix	20 Gaussians	
	Vehicle model (%)	Person model (%)
Vehicle data	93	7
People data	9	91

$$p(\lambda|\mathrm{O}) \propto p(\mathrm{O}|\lambda)P(\omega|a_T,\dots,a_1). \tag{7}$$

Using this rule with the 20 Gaussian HMM generates the following scatter matrix which represents a significant improvement in classification accuracy (Table 6).

Distance between models: In addition to determining the ability of the approach to correctly identify test data, the *distance* between the models may also be computed given the same set of tracks. This distance, which can also be interpreted as a measurement of the cross-entropy between the models [40], is calculated as follows for each observation set.

$$D_p(\mathrm{O}_p,\lambda_v,\lambda_p) = (\log P(\mathrm{O}_p|\lambda_v) - \log P(\mathrm{O}_p|\lambda_p)),$$

$$D_v(\mathrm{O}_v,\lambda_p,\lambda_v) = (\log P(\mathrm{O}_v|\lambda_p) - \log P(\mathrm{O}_p|\lambda_p)),$$

Table 7
Inter-model differences

	Number of Gaussians		
	5	10	20
Inter-model distance, Δ	0.5164	10.740	12.75

where λ_v and λ_p represent the vehicle and people models, respectively, and O_v and O_p are trajectory observations from the test set. These values essentially measure the ratio of the probability of a data set being explained by the incorrect model to the probability of the correct model. The expected value (mean) for both distance measures can be calculated by summing over all tracks as follows:

$$D'_p = \frac{1}{N_p}\sum_k D_p(\mathrm{O}^k_p, \lambda_v, \lambda_p),$$

$$D'_v = \frac{1}{N_v}\sum_k D_v(\mathrm{O}^k_v, \lambda_p, \lambda_v), \tag{8}$$

where O^k_p is the kth trajectory from the person observation set whose size is N_p. Similarly O^k_v is the kth trajectory from the vehicle observation set. In order to provide a symmetrical measure, D'_p and D'_v are combined as follows:

$$\Delta = \frac{1}{2}[D'_p + D'_v].$$

Values for Δ are presented in Table 7 for the 5, 10 and 20 state models. The higher level of mis-classification found in the previous section is supported by the low separation distance between the person and vehicle models.

6. Conclusions

The paper has presented design criteria and methodology for a distributed architecture for the interpretation of a dynamic scene. Computer vision and artificial intelligence techniques have been proposed to handle the distributed nature of the application and the uncertain nature of the acquisition process. Modular software has been suggested as the most suitable solution to the fusion of asynchronous information and the incremental creation of an event model able to capture scene dynamics and allow the classification of typical scene events.

6.1. Scene interpretation based on artificial intelligence

Latest trends in artificial intelligence, based on new software technology, have injected fresh hope in the computer vision community. The old days spent on the interpretation of a single static image are now only a distant past. On one hand artificial intelligence techniques based on the graphical stochastic models, such as Bayesian networks and Markov chains have been proven very successful at interpreting the dynamics of more or less cluttered scenes. On the other hand agent technology, based on efficient distributed systems has demonstrated the ability to process a large amount of information in parallel. Distributed systems are more reliable, easier to update and maintain. The distributed nature of the monitoring system, based on cameras scattered across large areas lends itself very well as an artificial intelligence system. Intelligence can be distributed across the monitoring system towards the cameras and sensors, to identify objects and track only those deemed of some interest. Intelligence can also be distributed towards processing units closer to the sensors to analyse locally the information gathered by the cameras, and broadcast only that part of some interest to the user. Because of the very nature of the application, communication between intelligent and autonomous agents becomes a critical issue. The proposed methodology proposes the use of agents to control the cameras and the objects created by events detected in the field of view of the cameras. Designing a communication protocol between agents entails a thorough study of the spectrum of possible modalities of communication, and the level of abstraction of communication.

6.2. The importance of stochastic methods

Video data are very challenging to analyse. They are inherently uncertain, incomplete, and data streams—acquired by sensors and cameras—are intrinsically asynchronous. Stochastic models such as Bayesian networks and Markov chains have enjoyed a large amount of success in the last decade. Such models are formally proved and can be easily learned, updated and used to classify new accrued information about scene dynamics. Stochastic models are ideal to build the interpretation of the scene, in terms of stochastic processes—the events—which might or might not be statistically related. Stochastic models provide an ideal method of bridging the gap between raw data and semantic information about the underlying analysed process. Both Bayesian networks and Markov chains endow the agent with reasoning power by means of an inference mechanism solidly based on probabilistic calculus. Stochastic models lend themselves very well to an efficient encoding of information, allowing the agent to broadcast the required information in a fast and compact way. Finally stochastic models can be easily combined to produce more complex descriptions of the evolution of a scene.

References

[1] P. Remagnino, G.A. Jones, N. Paragios, C.S. Regazzoni (Eds.), Video-Based Surveillance Systems: Computer Vision and Distributed Processing, Kluwer Academic Publishers, Dordrecht, 2002, ISBN/ISSN 0-7923-7632-3.

[2] G.L. Foresti, C.S. Regazzoni, P.K. Varshney, Multisensor Surveillance Systems: the fusion perspective. Kluwer Academic Publishers, July 2003.

[3] R.W. Picard, Affective Computing, MIT Press, Moscow, 1997.

[4] E. Horvitz, Principles of mixed-initiative user interfaces, Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems, Pittsburgh, PA, May 1999.

[5] H. Van Dyke Parunak, Applications of distributed artificial intelligence in industry, in: G.M.P. O' Hare, N.R. Jennings (Eds.), Foundations of Distributed Artificial Intelligence, Wiley Interscience, New York, 1996, pp. 139–164.

[6] K.S. Decker, Distributed problem-solving techniques: a survey, IEEE Trans. Syst., Man Cybern. SMC-17 (5) (1987) 729–740.

[7] K. Sycara, K. Decker, A. Pannu, M. Williamson, D. Zeng, Distributed intelligent agents, IEEE Expert 11 (6) (1996) 36–46.

[8] M. Wooldridge, Intelligent Agents, in: G. Weiss (Ed.), Multiagent Systems A Modern Approach to Distributed Artificial Intelligence, MIT Press, Cambridge, Moscow, 1999, pp. 27–77.

[9] Y. Shoham, Agent-oriented programming, Artif. Intell. 60 (1993) 51–92.

[10] K.P. Sycara, Multiagent systems, AI Magazine 10 (2) (1998) 79–93.

[11] P. Stone, M. Veloso, Multiagent systems: a survey from a machine learning perspective, Auton. Robots. 8 (2000) 345–383.

[12] J. Ferber, Multi-agent systems, An Introduction to Distributed Artificial Intelligence, Addison-Wesley, Reading, MA, 1999.

[13] C. Hewitt, Viewing control structures as patterns of passing messages, Artif. Intell. 8 (1977) 323–364.

[14] G. Booch, Object-Oriented Analysis and Design with Applications, Benjamin/Cummings, 1994.

[15] S.S. Intille, A.F. Bobick, A framework for recognizing multi-agent action from visual evidence, in: Proceedings of the 16th National Conference on Artificial Intelligence, Orlando, Florida, July 1999, pp. 518–525.

[16] P. Remagnino, T. Tan, K. Baker, Agent orientated annotation in model based visual surveillance, in: Proceedings of IEEE International Conference on Computer Vision, Mumbai, India, January 1998, pp. 857–862.

[17] H. Buxton, S. Gong, Visual surveillance in a dynamic and uncertain world, Artif. Intell. 78 (1995) 431–459.

[18] B. Rosario, N. Oliver, A. Pentland, A synthetic agent system for Bayesian modeling of human interactions, in: Proceedings of Conference on Autonomous Agents, May 1999, pp. 342–343.

[19] A.D. Wilson, A.F. Bobick, Parametric hidden Markov models for gesture recognition, IEEE Trans. Pattern Anal. Mach. Intell. 21 (9) (1999) 884–900.

[20] M. Brand, Structure learning in conditional probability models via an entropic prior and parameter extinction, Neural Comput. 11 (1999) 1155–1182.

[21] J. Pearl, Probabilistic Reasoning Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, Los Altos, CA, 1988.

[22] J. Pearl, Causality, Models Reasoning and Inference, Cambridge University Press, Cambridge, 2000.

[23] L. Rabiner, B-H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[24] A. Papoulis, Probability, Random Variables, and Stochastic Processes, McGraw-Hill, New York, 1984.

[25] J. Yamato, J. Ohya, K. Ishii, Recognising human action in time-sequential images using hidden Markov models, in: Proceedings of Conference on Computer Vision and Pattern Recognition, June 1992, pp. 379–385.

[26] M. Brand, N. Oliver, A. Pentland, Coupled hidden Markov models for complex action recognition, Proceedings, CVPR, IEEE Press, New York, 1997, pp. 994–999.

[27] L. Marchesotti, Development of distributed agent-based architecture for distributed video surveillance applications, Ph.D. Thesis, DIBE, University of Genoa, 2001.

[28] P. Remagnino, T. Tan, K. Baker, Multi-agent visual surveillance of dynamic scenes, in: Image and Vision Computing, 16, Elsevier Science, Amsterdam, 1998, pp. 529–532.

[29] P. Remagnino, T. Tan, K. Baker, Agent orientated annotation in model based visual surveillance, Proceedings of the International Conference in Computer Vision, 4–7 January, Mumbai, India, 1998, pp. 857–862.

[30] J.G. Semple, G.T. Kneebone, Algebraic projective geometry, Oxford University Press, Oxford, 1998.

[31] G.A. Jones, J.R. Renno, P. Remagnino, Auto-calibration in multiple-camera surveillance environments, Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, June 1, Copenhagen, Denmark, 2002, pp. 40–47.

[32] C. Stauffer, W.E.L. Grimson, Learning patterns of activity using real-time tracking, IEEE Trans. Pattern Anal Mach. Intell. 22 (8) (2000) 747–757.

[33] A. Bakowski, G.A. Jones, Video surveillance tracking using colour adjacency graphs, IEE Conference on Image Processing and its Applications, 1999, pp. 794–798.

[34] P. Remagnino, S. Maybank, R. Fraile, K. Baker, R. Morris, Automatic visual surveillance of vehicles and people, in: C.S. Regazzoni, G. Fabri, G. Vernazza (Eds.), Advanced Video-based Surveillance Systems, Kluwer, Dordrecht, 1998, pp. 97–107.

[35] J. Orwell, P. Remagnino, G.A. Jones, From connected components to object sequences, First IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, 2000, pp. 72–79.

[36] D.H. Ballard, C.M. Brown, Computer Vision, Prentice-Hall, Inc., New Jersey, 1982.

[37] P. Remagnino, J. Orwell, D. Greenhill, G.A. Jones, L. Marchesotti, An agent society for scene interpretation, in: G. Foresti, P. Mahonen, C.S. Regazzoni (Eds.), Multimedia Video-based Surveillance Systems, Requirements, Issues and Solutions, Kluwer, Dordrecht, 2000, pp. 108–117.

[38] A. Dempster, M. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. B 39 (1977) 1–38.

[39] V. Cadez, S. Gaffney, P. Smyth, A general probabilistic framework for clustering individuals and objects, in: Proceedings of the ACM KDD, August 2000.

[40] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley Interscience, New York, 1991.

[41] R.Y. Tsai, A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, IEEE J. Robot. Automation RA-3 (4) (1987) 323–344.

About the Author—DR. PAOLO REMAGNINO joined the Digital Imaging Research Centre at Kingston University in September 1998. He is an active researcher in the field of Machine Learning and Computer Vision and his expertise encompasses the Computer Vision and Artificial Intelligence research fields, specialising in modern and future methodologies and technology for the design of ambient intelligence for advanced monitoring of open and closed environments. Dr. Remagnino has been an active researcher for more than a decade, and during the last five years he worked to build and maintain a European community of experts in Visual Surveillance. He has participated and coordinated major Visual Surveillance events and he is the main editor of a book on Visual Surveillance. Dr. Remagnino's current research interest focuses on improving existing visual surveillance techniques to develop autonomous intelligent environments, promoting the design and development of Ambient Intelligence.

About the Author—DR. AHMED I. SHIHAB joined the Digital Imaging Research Centre at Kingston University in August 2001. He received his B.Sc. degree from the Department of Computers and Automatic Control, Ain Shams University, Cairo, and received his Ph.D. degree from the Department of Computing, Imperial College, London. Dr. Shihab's research interests are in data clustering and in modelling coordinated activities, particularly in sports video.

About the Author—DR GRAEME A. JONES is the Director of Digital Imaging Research Centre at Kingston University. Dr. Graeme A. Jones has over 15 years of experience in image and video sequence analysis and has authored or co-authored over 100 technical papers related to computer vision and video data communications. A graduate of Edinburgh University, Dr. Jones received his Ph.D. in Computer Vision from the University of London in 1994. Since 1997 he has been a Reader in Computer Vision at Kingston University. He is currently a co-investigator on an EPSRC grant (GR/N17706/01) on traffic simulation and optimisation of video surveillance networks, a co-investigator on the EU INMOVE project (IST-2000-37422) on intelligent mobile video environments, and principle investigator on the Video-based Threat Assessment and Biometrics Network GR/S64301/01 which aims to encourage the maturation of those video-interpretation technologies most likely to enhance future police identification, authentication and threat assessment capabilities. In 2000, Dr. Jones chaired the British Machine Vision Association workshop on Visual Surveillance and was co-chair of the IAPR Workshop on Advanced Video-based Surveillance Systems in 2001.