A Stochastic Approach to Tracking Objects Across Multiple Cameras

Anthony R. Dick and Michael J. Brooks

School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia
CRC for Sensor, Signal and Information Processing, Technology Park,
Mawson Lakes, SA 5095
{ard, mjb}@cs.adelaide.edu.au

Abstract. This paper is about tracking people in real-time as they move through the non-overlapping fields of view of multiple video cameras. The paper builds upon existing methods for tracking moving objects in a single camera. The key extension is the use of a stochastic transition matrix to describe people's observed patterns of motion both within and between fields of view. The parameters of the model for a particular environment are learnt simply by observing a person moving about in that environment. No knowledge of the environment or the configuration of the cameras is required.

1 Introduction

Tracking moving objects as they are observed by a video camera is a much researched problem in computer vision. Although it is far from solved, there is a variety of existing systems that work quite reliably under certain well documented conditions. A thorough review is beyond the scope of this paper, but some examples are Boujou[1], which tracks point features, Condensation [8] and its variants, which track image contours, and mean-shift methods [2] which track colours. Other tracking systems are specialised to tracking people [1] or certain objects whose shape is known in advance [5].

Many applications, however, demand tracking over a wider area than can be covered by a single camera. For instance, a building may be monitored by a network of surveillance cameras, and ideally a person would be tracked consistently as they move between them [3]. One solution is implement blanket coverage of the building, so that every part of it is visible to at least one camera. If the cameras are calibrated, and the structure of the environment is known, then people will pass from one camera's field of view (FOV) to another at known locations, which act as trigger points for handing over tracking from one camera to the next. This however is impractical for an environment of any size or complexity, due to the number of cameras required and physical constraints on their placement.

[1] www.2d3.com

Tracking algorithms will therefore have to deal with "blind spots" where a person is not visible to any camera if they are to operate on a network of cameras monitoring a complex environment. Most tracking algorithms in the computer vision literature are not well suited to this task, because typically they rely on the assumption of smooth motion, using previously observed velocity to predict the future location of their target. An example of this is the Kalman filter [14], and various types of particle filter [8, 4], which are more robust than the Kalman filter to sudden changes but are based on the same underlying assumption. When tracking an object between non-overlapping fields of view, its motion is almost guaranteed not to be smooth. For instance, a person may disappear from the right side of one FOV and then reappear after some time at the top of another FOV, depending on how the cameras are positioned.

This problem has become increasingly apparent in recent years, and has been the subject of some recent research in computer vision. Kettnaker [10] presents a Bayesian approach to tracking objects through FOVs that do not overlap. However this system requires a set of allowable paths, and a set of transition probabilities and their expected duration, to be supplied a priori. This effectively requires that the environment is known in advance, along with the ways in which people move about in it, and the positions of the cameras. In many surveillance applications, this information is difficult or impossible to obtain.

A traffic monitoring system using cameras situated about 2 miles apart along a highway is described in [7, 11]. Cars are identified as they enter each camera's field of view based on both their appearance and positional information from views through which they have previously passed. The surveillance of traffic is well suited to cameras with non-overlapping FOVs, because traffic generally follows well defined paths which extend over long distances.

Recently, work has been carried out on tracking people or objects with more general motion between non-overlapping camera FOVs. This has been approached as a problem of learning the probability of transitions between fields of view from corresponding tracks (e.g. see [9]). However, the correspondence between tracks in different images must be supplied a priori, as training data. Ellis et al. [6] do not require correspondences, instead observing motion over a long period of time and recording each appearance and disappearance. All possible pairings between appearance and disappearance are accumulated in a histogram, and those which occur most consistently are considered the true transitions.

In this paper we describe a system for tracking objects across the FOVs of cameras that do not overlap. The system does not require any knowledge of the cameras' placement or calibration, nor does it need to know anything about the environment it is monitoring. It works by learning a Markov model describing the paths taken by a marker as it is carried around in the environment.

The structure of this paper is as follows. In Section 2, we review a single camera tracking technique which we use as part of our multiple camera tracker. In Section 3, we introduce the Markov model we use to model the motion of objects within and between cameras, and describe how this model is trained for a particular camera setup and environment. Following this the Markov model is combined with the single view tracker to form our multiple view tracking algorithm, described in Section 4. Some illustrative results of the tracker are then presented in Section 5.

2 Tracking Objects with Background Subtraction

Our method for tracking people across multiple cameras is built on an existing technique for tracking within a single camera's field of view. In particular, we implemented the multimodal background subtraction algorithm of Stauffer and Grimson [13].

Background subtraction is a method for object detection that maintains a model of the colour of the background at each pixel; in [13], the model is a mixture of Gaussians. In each frame of video, the likelihood of the current colour of each pixel is computed according to its background model. Based on this likelihood, each pixel is classified as foreground or background. Foreground pixels are those which are not well explained by the background model, and are therefore likely to be interesting objects to track. Clusters of foreground pixels are grouped into objects which can then be tracked as a whole. Stauffer et al. use a Kalman filter to track these objects across frames.

The problem with a Kalman filter is that motion observed in previous frames is used to predict the motion in the current frame. This assumption holds for an object moving smoothly in a single camera's field of view, in an uncluttered environment. However as objects move between camera FOVs, there will inevitably be a discontinuity in their observed motion. Even within a single FOV, if the target object moves behind an obstacle which hides it from view and then reappears at another point in the image, its observed motion is no longer continuous.

We therefore need a tracking algorithm that can cope with discontinuities in order to extend this method to track objects across multiple FOVs. Our algorithm is based on a Markov model, which is introduced in the next section.

3 The Markov Model

We model the movement of each object as a Markov process. A Markov process (or model) is one which at time t can be in any one of N states $S = \{s_1 ... s_N\}$. The way it progresses between these states is governed by:

- an N dimensional initial state probability vector π, and
- a stochastic $N \times N$ *transition matrix* A, whose (i, j)th element defines the probability of moving from state s_i to s_j.

To cast our tracking system as a Markov model, each camera's field of view is divided evenly into $G \times G$ pixel squares, each of which is associated with a state. The square in which the object currently appears is the current state of the model.

In this context, each element A_{ij} of the transition matrix defines the probability that the object moves from the image region associated with state S_i at frame t to that associated with state S_j at frame $t+1$ (A is stationary). Each coefficient π_i denotes the probability that an object appears for the first time in the region associated with S_i.

This is a very general model of motion, allowing an object to move instantly from any part of any camera FOV to any other. Of course in practice objects do not behave this way. The transition matrix and the initialisation vector need to be assigned values that reflect the way people or objects actually move in the environment under observation.

3.1 Training the Markov Model

We learn appropriate values for A and π from a series of observations. The observations are generated by a person carrying an easily identifiable marker as they move around in the environment. The marker (we used a bright red soccer ball, see Figure 1) is used to ensure the person is tracked reliably, even when there is more than one person moving about in the environment. This was found to be more practical than the alternative, which was to clear the environment of other people and track a single person as they move about.

The marker is tracked simply by finding the image patch whose redness relative to its usual background colour is the highest. If no image patch appears significantly redder than usual, it is assumed that the marker is not currently visible. In practice, this turned out to be very reliable in our rather beige office environment. The training program was left to run in the background for hours on end, without picking up false tracks.

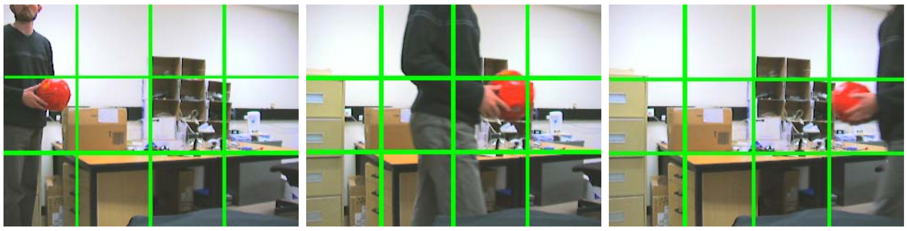


Fig. 1. Training the Markov model. Each 80 by 80 pixel square is assigned a state. Each transition of the red ball between states (the image regions bordered by grid lines, shown here in green) is recorded

To learn values for A and π, we use a simple occurrence counting method. For each frame, as the marker moves between states S_i and S_j in the camera views, the value of A_{ij} is incremented. Similarly, each time a new track is initialised in state S_i, the corresponding value π_i is incremented. Training does not have to be carried out continuously. It was common during experimentation to train the model for a few minutes and then save the current values of A and π, to be loaded up and further refined later on. To prevent A from being dominated by entries in its leading diagonal, which correspond to the marker remaining in the same state, A_{ii} is not incremented if the marker is detected in the same state

S_i in consecutive frames. A_{ii} is incremented, however, if the object appears in state S_i, then disappears for one or more frames, and then reappears in state S_i. After training, A is normalised so that each row sums to 1. π is also normalised so that its entries sum to 1.

This method is analogous to the well known Baum-Welch algorithm [12] for Hidden Markov Models (HMMs), when the states are not hidden (i.e. one can derive the current state of the model from the current observation). The removal of the level of inference that "hides" the states from the observations means that considerably less training data is required for our Markov model than would be needed for the equivalent HMM.

4 Object Tracking

We now apply the Markov model described in the previous section to the results of the object detection method described in Section 2. Recall that having located foreground pixels and connected them to form objects, we have some number of objects detected in the field of view of each camera. These objects are now matched with corresponding objects detected in the previous frame of video.

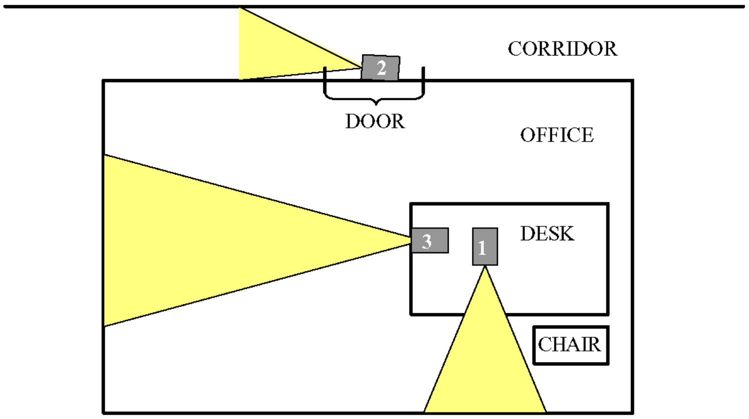
In practice, it was useful to use Kalman filters as a first pass tracking mechanism. The Kalman filter and the Markov model are complementary, in the sense that the Kalman filter, having a continuous state vector, is better suited to resolving short tracks between frames, whereas the Markov model, having discrete states of $G \times G$ pixels, is better at tracking fast motion or motion that the Kalman filter cannot predict. Thus the Kalman filter will usually pick up objects that have moved slowly and smoothly within the field of view of a single camera, while the remaining objects are then matched using the Markov model, as follows.

First, the current state S_C of each unmatched object is computed by working out the location of its centre. We then consider all objects that disappeared between 2 and K frames ago as candidate matches. The object that disappeared in the location whose state S_D maximises the transition probability A_{DC} is matched with the currently visible object. If the maximum value of A_{DC} is less than the probability that a new track has begun in this location, π_C, the object remains unmatched and is labelled as being previously unseen.

The set of matches that maximise each individual probability may not be the set of matches that maximises the combined probability of all matches. Whenever the probabilities of multiple appearances are maximised by the same disappearance, the matching process is repeated assigning that disappearance to each appearance in turn. The assignment which maximises the combined probability is retained.

5 Results and Observations

We consider a modest network of 3 cameras, all attached to the same desktop PC. The approximate layout of the cameras is shown in Figure 2. Each camera delivers a 320×240 video stream at about 15 frames per second.



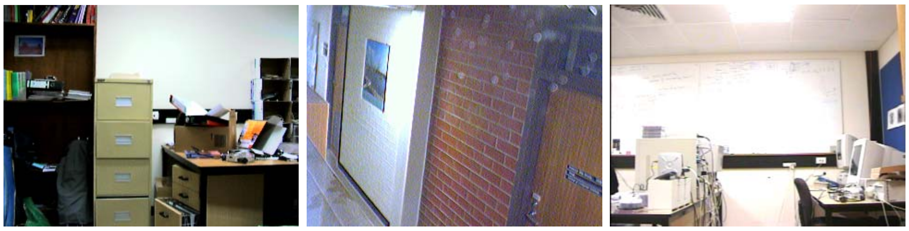


Fig. 2. Top: Camera network layout. Cameras 1 and 3 are on my desk, pointing at right angles to each other. Camera 2 is positioned above the door looking down the corridor outside my office. Bottom: View from cameras 1, 2 and 3

The Markov model was trained sporadically over a period of weeks using the method described in Section 3.1, by the authors and other volunteers carrying the ball around in the office and the corridor outside. The same model parameters were used for all experiments described in this paper. Each state in our model is associated with an 80×80 square of pixels, meaning that there is a total of 12 states per camera and 36 states overall. Fortunately, motion in our environment is quite constrained, so although the transition matrix contains 36×36 entries, far fewer than that need to be learnt.

Our first set of experiments involves tracking a single person ("Person A") as he moves through this environment, within and between the cameras' fields of view. The sequence of movements is as follows: Person A gets out of his desk chair, then moves right to left through the field of view of camera 1 (Fig 3(a)) to the far side of the office, in the field of view of camera 3 (Fig 3(b)). Not satisfied with this, he then leaves the office and walks down the corridor to an adjacent office. This takes him through the field of view of camera 2, from bottom to top (Fig 3(c),(f),(g)). He then re-enters the office and returns to his chair. This path takes him through the fields of view of all cameras and several blind spots between them.

While this is taking place other people are walking along the corridor outside the office (Fig 3(d),(e)). The main challenges in this experiment are to correctly

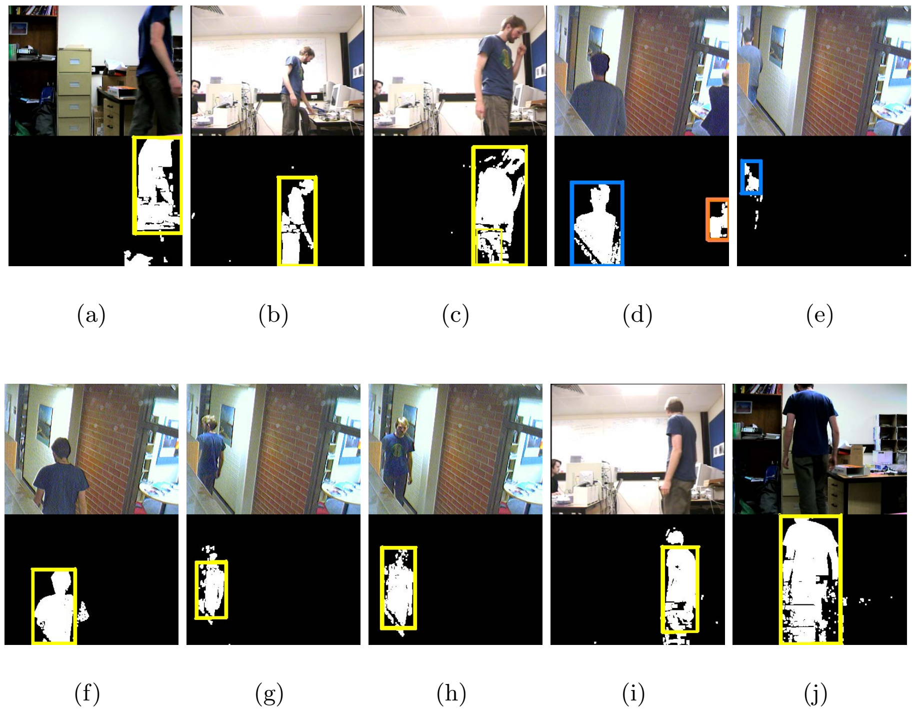


Fig. 3. Snapshots (in order of time) from tracking experiment 1. Each snapshot consists of the video frame (top) and the tracking result (bottom). The subject is successfully tracked through all camera views despite the presence of distractions. Passers-by in the corridor ((d) and (e)), are identified as different people (marked with different colour bounding box), while the subject is identified consistently across all views (same colour) despite disappearing and reappearing several times

link each reappearance of Person A with his disappearance from the previous view, and to track each passerby in the corridor while recognising that each one is the appearance of a new, previously unseen person.

Snapshots of the experiment are shown in Figure 3, in order of time. In each snapshot, the image captured by the relevant camera is shown above the results of the object detection and tracking algorithm. The white pixels in these results are those classified as foreground. Rectangles are drawn around those groups of foreground pixels that are detected as objects and matched to objects in a previous frame. Each object is identified by the colour of its surrounding rectangle. From these results it can be seen that Person A (yellow rectangle) is successfully tracked between views as he moves about in his environment. An example is shown of passers-by who are tracked in the corridor, while Person A is still in his office. The track is successfully maintained for both person A and this passerby.

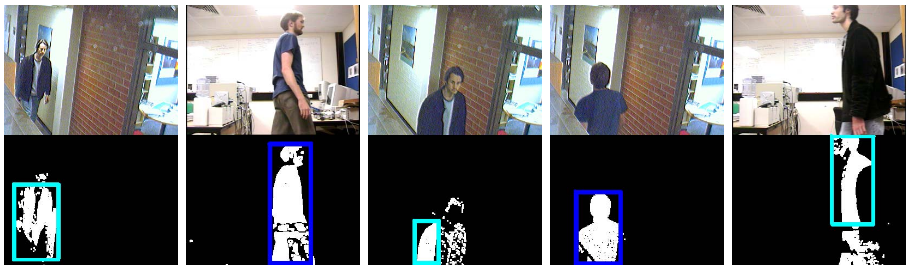


Fig. 4. Snapshots from tracking experiment 2, involving 2 people crossing in the office doorway, out of view of the cameras. The tracks are correctly maintained

A second experiment involves 2 people moving in opposite directions through the camera FOVs. Person A moves from his desk, out of the office and into the corridor. Meanwhile person B moves from an adjacent office, down the corridor and into the office of person A. Persons A and B cross over at the doorway to the office, which is not visible to any camera. The tracker therefore relies on the Markov model to make the correct decision when person A appears in camera 3 and person B appears in camera 2 (the corridor camera), after they cross over. In both cases the tracks are maintained correctly, as shown in Figure 4.

Experiment 3 involves a similar situation, with 2 people moving about in the environment simultaneously. This time, person A leaves his office (Fig 5(a)) and heads down the corridor (Fig 5(c)), exiting at the top of the FOV of the corridor camera. Person B follows A (Fig 5(b),(d)), into the corridor then exits to the bottom of the corridor camera (Fig 5(e)). Persons A and B then reappear and return to their desks (Fig 5(f),(g),(h)).

This sequence shows the ability of the Markov model to decide which disappearance best explains each appearance, based on the location of each. Having been trained, the model knows that it is unlikely to observe a person disappear from the top of the corridor camera and reappear at the bottom, and vice versa.

Experiment 4 shows how the tracker can be misled. Person A is tracked correctly to the door of his office (Fig 6(a),(d)), while at the same time person B exits the office across the corridor (Fig 6(b),(c)). Person B is correctly identified as a new appearance, and then exits to the bottom of the FOV of the corridor camera. Person A then enters the corridor camera from the bottom, and is mistakenly identified as a reappearance of person B (Fig 6(e)). This reflects the Markov model's belief that an appearance at the bottom of corridor camera is more likely to be someone walking along the corridor than exiting the office. This belief is quite reasonable—more people are likely to observed walking along the corridor than exiting person A's office—but wrong in this case.

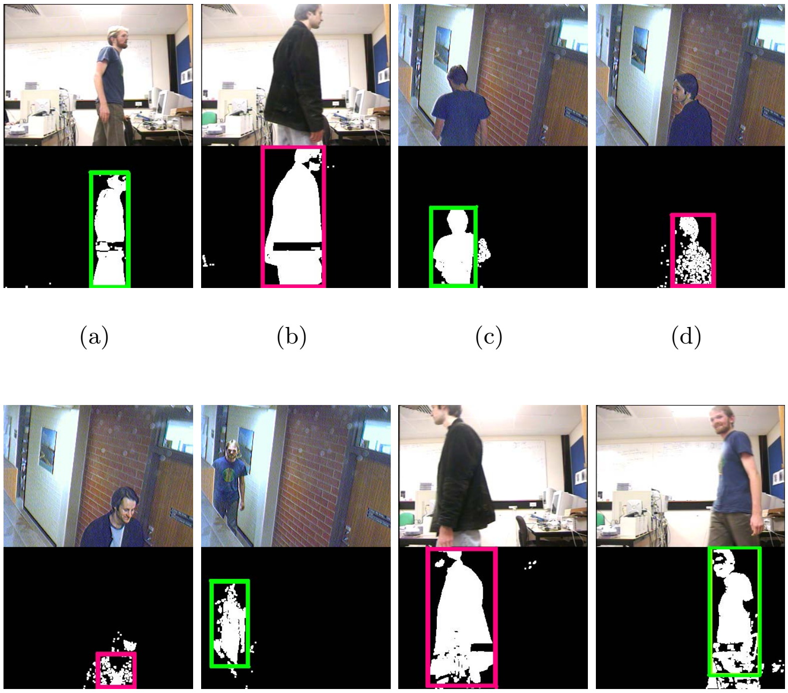


Fig. 5. Person B follows person A out of the office, then turns around. Person A then returns after person B. Both people are correctly identified, despite the fact that person B disappears and reappears. Again, frames are shown in order of time

5.1 Observations

Given the failure case in Experiment 4, it seems like it might be useful to use the appearance of each object to help identify it. However, it was found that due to the difference in lighting, camera characteristics, and their points of view, the appearance of an object could change significantly between cameras. This means that appearance can not be used in a straightforward way to disambiguate appearances and disappearances in camera FOVs. Recognising this, Javed et al. [9] learn the way appearance changes between cameras rather than using similarity of appearance to match objects between views. We intend to investigate similar approaches.

This system operates in real time, at around 5-10 frames per second. The bottleneck is the bandwidth required to capture 3 streams of video on a PC, which could be overcome using specialised capture hardware or by processing the streams on different machines. Computationally the most expensive procedure is the multimodal background subtraction update; in fact once learnt, the Markov model takes very little compute time to apply.

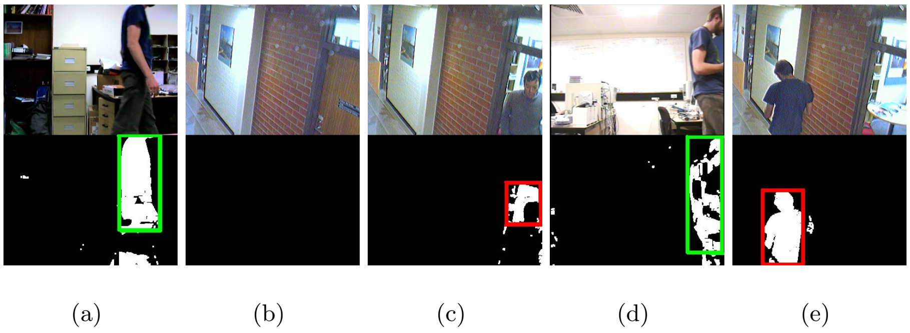


Fig. 6. A case in which the tracker fails. Person A in the blue shirt is tracked correctly to the door of his office. During this time person B leaves his office, and exits to the bottom of the FOV of camera 2. As person A enters the FOV of camera 1 from below, he is assigned the identity of person B

6 Conclusion and Future Work

In this paper we have introduced a real-time algorithm that uses a Markov model to track objects as they move between the separated FOVs of multiple cameras. We have described the way in which the parameters of the Markov model are learnt and the way it is applied to an existing single camera tracking algorithm.The feasibility of the algorithm has been demonstrated using several scenarios in our office camera network.

There are a number of extensions that could be made to this work. We could extend the Markov model to a Hidden Markov model that combines its motion model probabilistically with the output of the tracker, which would also be expressed probabilistically. It would also be useful to maximise the probability of the observed tracks over the previous N frames rather just the previous 1 frame. The Viterbi algorithm [12] is a well known method for doing this for HMMs. One drawback of Markov models, hidden or not, is that transitions between states occur instantaneously. We plan to incorporate information about transition times by storing a histogram for each transition of the durations observed for that transition during the training phase. This can then be combined with the overall transition probability to disambiguate track discontinuities based on timing.

References

1. J.K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, March 1999.
2. D. Comaniciu and V. Ramesh. Mean shift and optimal prediction for efficient object tracking. In *ICIP00*, pages Vol III: 70–73, 2000.
3. A.R. Dick and M. J. Brooks. Issues in automated video surveillance. In *Proc. 7th International Conference on Digital Image Computing: Techniques and Applications (DICTA'03)*, pages I:195–204, Sydney, 2003.

4. A. Doucet, N. de Freitas, N. Gordon, and eds. *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, 2001.
5. T. Drummond and R. Cipolla. Application of lie algebras to visual servoing. *International Journal of Computer Vision*, 37(1):21–41, 2000.
6. T. J. Ellis, D. Makris, and J.K. Black. Learning a multi-camera topology. In *Joint IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 165–171, 2003.
7. T. Huang and S. Russell. Object identification in a Bayesian context. In *Proceedings of IJCAI*, pages 1276–1283, 1997.
8. M. Isard and A. Blake. Condensation–conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
9. O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking across multiple cameras with disjoint views. In *Proc. IEEE International Conference on Computer Vision*, pages 952–957, 2003.
10. V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 253–259, 1999.
11. Hanna Pasula, Stuart J. Russell, Michael Ostland, and Yaacov Ritov. Tracking many objects with many sensors. In *Proceedings of IJCAI*, pages 1160–1171, 1999.
12. L. R. Rabiner. A tutorial on hidden markov models and selected apllications in speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 267–296. Kaufmann, San Mateo, CA, 1990.
13. C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.
14. G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report 95-041, University of North Carolina at Chapel Hill, 1995.