Tennis Ball Tracking for Automatic Annotation of Broadcast Tennis Video

Fei Yan

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

June 2007

Summary

This thesis describes several algorithms for tracking tennis balls in broadcast tennis videos. The algorithms are used to provide a ball tracking module for an automatic tennis video annotation system.

As an overall strategy, we have decided to adopt the Track-After-Detection (TAD) approach to the tracking problem. A TAD approach decomposes a tracking problem into two sub-problems: object candidate detection and object candidate tracking. In the second sub-problem, resolving data association ambiguity is the key. In this thesis, we propose a novel tennis ball candidate detection algorithm, and three novel data association algorithms.

The ball candidate detection algorithm extracts foreground moving blobs using temporal differencing, and classify the blobs into ball candidates and non-candidates. A novel gradient-based feature for measuring the departure of the shape of a blob from an ellipse is proposed.

The proposed data association algorithms take as input the positions of the ball candidates, and try to resolve the object-candidate association ambiguity. The first data association algorithm is based on a particle filter with an improved sampling scheme. Smoothing and an explicit data association process are also used to increase the accuracy of the tracking results. The second data association algorithm is based on the Viterbi algorithm. It seeks the sequence of object-candidate associations with maximum *a posteriori* probability, using a modified version of the Viterbi algorithm. The third data association algorithm is a multi-layered one with graph-theoretic formulation. At the bottom layer, "tracklets" are "grown" from ball candidates with an efficient variant of the RANdom SAmple Consensus (RANSAC) algorithm. The association problem is then formulated as an all-pairs shortest path (APSP) problem in a graph with tracklets as its nodes, and is solved with a novel fast APSP algorithm.

The proposed data association algorithms have been tested using tennis sequences from the Australian Open tournaments. Their performances are compared both mutually and to existing object tracking algorithms. Experiments show that the particle filter based algorithm gives similar performance to the state-of-the-art tennis ball tracking method in the literature: the Robust Data Association (RDA), while the other two proposed algorithms both outperform RDA.

Key words: Sports Video Annotation, Tennis Ball Tracking, Data Association, Sequential Monte-Carlo Method, Viterbi Algorithm, Graph Theory, Shortest Path

Email: f.yan@surrey.ac.uk

WWW: http://www.ee.surrey.ac.uk/cvssp

Acknowledgements

I would like to thank my supervisors, Dr. William Christmas and Prof. Josef Kittler, for their guidance and support throughout the course of this work. I would also like to thank everyone who has offered his or her help, encouragement and friendship. In particular, thanks are due to Yonggang Jin and my colleagues in the VAMPIRE team, Ilias Kolonias and Alexey Kostin, for the stimulating discussions.

This work was supported under the following projects:

EU IST-2001-34401 Project VAMPIRE

EU IST-507752 MUSCLE Network of Excellence

Contents

Chapter 1

Introduction

1.1 Motivation

The rapid growth of sports video database demands efficient tools for automatic sports video annotation. Owing to recent advances in both computer hardware and computer vision algorithms, building such tools has become possible. The sports being annotated include soccer [79], Formula-1 racing [49], snooker [18], tennis [16, 33]. Such sports video annotation systems have many potential applications, *e.g.* video retrieval, enhanced broadcast, object-based video encoding, video streaming, analysis of player tactics, computer aided coaching.

In the past three years, my colleagues and I have been building a tennis video annotation system. That is, a computer vision system that can "understand" tennis, in the sense that it takes as input a broadcast tennis video, and generates high level annotation, such as the score of the game, as the game progresses [6, 32, 35].

In automatic video annotation, high level descriptions rely on low level features. In the context of a tennis game, the ball trajectory contains a rich set of information for the annotation. The motivation of the work presented in this thesis was to provide a tennis ball tracking module for our tennis video annotation system. The reconstructed ball trajectories given by the ball tracking module are used together with other low level features, such as player trajectories and court lines, to generate high level annotation.

1.2 Difficulties of Ball Tracking in Broadcast Tennis Video

Tennis ball tracking in broadcast tennis video is a challenging task. A tennis ball has a very small size, and can travel at very high velocity. For example, in a frame with resolution 268×670, a tennis ball can be as small as only 5 pixels when it is at the far side of the court. A tennis ball can travel at up to 70 meters per second, which means in the image plan, the distance between its positions in two successive frames can be more than 20 pixels.

A tennis ball is also subject to motion deformation, motion blur, successive occlusion, and sudden change of motion direction. All these problems pose great challenges for the tracking, especially when they are combined. For example, when the ball is hit by a player, it changes its motion drastically. Since the ball travels at very high velocity after being hit, it is often blurred into background, and cannot be detected in the first few frames. As a result, the next detected ball position can be very far from the predicted position that is obtained using an obsolete motion model. This is much more challenging than tracking an object with a smooth manoeuvre. Many existing trackers would lose track in such a situation.

The uncontrollability of the making of tennis video adds further complexity. For example, there may be multiple plays in one sequence; there may be balls thrown in by ball boys when the ball used for play is still in the scene; the camera may not pan or tilt fast enough to keep up with the ball, as a result, the ball may exit the scene and then re-enter. In brief, tennis ball tracking in monocular sequences is a multiple object tracking problem where the objects can change their motions abruptly; new objects can appear; existing objects can disappear and reappear. It is too challenging for classical methods to work directly without much careful and pertinent redesign. Some examples of the difficulties of ball tracking in monocular tennis video are shown in Fig. 1.1 and Fig. 1.2.

1.3 The Overall Strategy

Object tracking is one of the key topics of computer vision. Briefly speaking, object tracking is concerned with finding and following the object of interest in a video sequence. Two pieces of information are essential for any object tracking problem: object appearance and object dynamics. All object tracking techniques somehow make use of both pieces of information. Depending

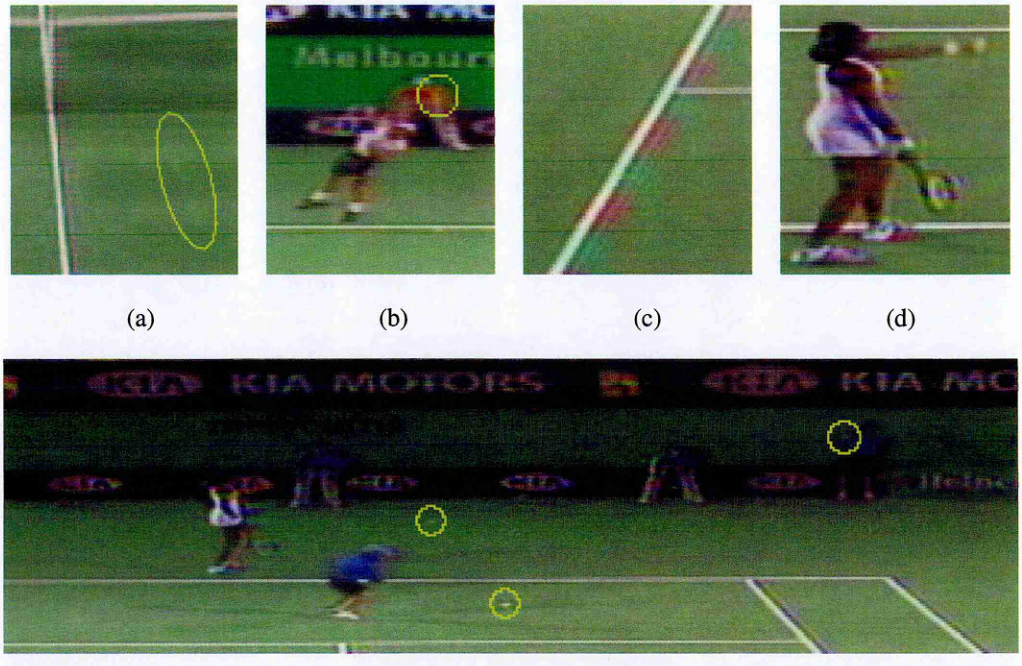
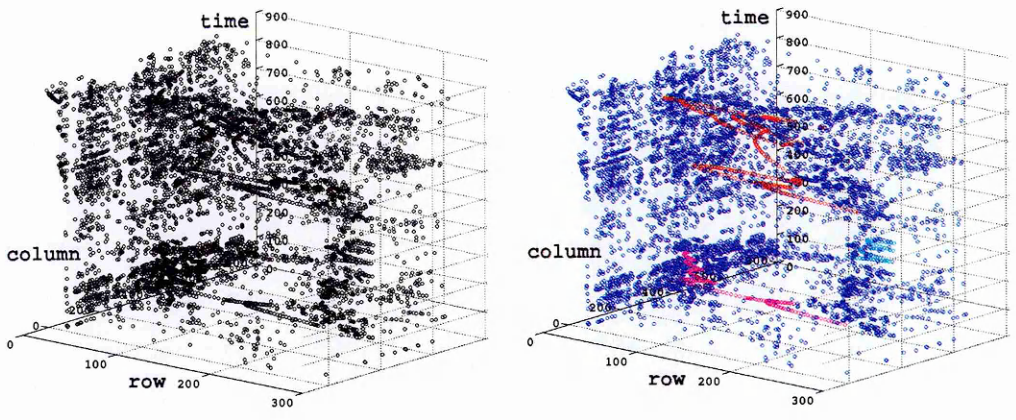


Figure 1.1: Images cropped from frames showing the difficulties of tennis ball tracking - part 1. (a) Fast moving ball. The ball is blurred into background and is very hard to detect. (b) Far player serving. The ball region in this frame contains only a few pixels, and its colour is strongly affected by the background colour. (c) Chroma noise caused by PAL cross-colour effect. This may introduce false ball candidates along the court lines. (d) A wristband can look very similar to the ball, and can form a smooth trajectory as the player strikes the ball. (e) Multiple balls in one frame. A new ball (left) is thrown in by a ball boy while the ball used for play (middle) is still in the scene. There is another ball (right) in the ball boy's hand.

Figure 1.2: Images cropped from frames showing the difficulties of tennis ball tracking - part 2. The sub-figures are cropped from 12 successive frames where a ball is hit by the near player. The combination of sudden change of tennis ball motion and motion blur can lead to loss of track. Order of the sub-figures: from left to right, top to bottom.



(a) (b)

Figure 1.3: In TAD, resolving data association ambiguity is the key. (a): All ball candidates in a sequence plotted in a column-row-time 3D space. Each black circle is a ball candidate. (b): After candidate detection, the key of candidate tracking is to recover the class labels of each candidates: clutter, first play, second play, or third play. Blue circles: clutter-originated candidates. Circles in other colors: ball-originated candidates. The magenta, cyan, and red circles correspond to the first, second and third play in the sequence, respectively.

on the the way these two pieces of information are combined, object tracking algorithms can be broadly categorised into two main approaches: Track After Detection (TAD) type of approach and Track Before Detection (TBD) type of approach.

.

In the TAD approach, object candidates are first detected in each frame. After this step, **only** the candidates are used for the tracking, the rest of the image is discarded. In such an approach, once object candidates are detected, the tracking problem involves two elements: data association and estimation. The first element, data association, deals with measurement origin uncertainty, *i.e.* which candidates are object-originated and which are clutter-originated (which candidates are which-object-originated in a multiple object tracking scenario); while the second element, estimation, deals with measurement inaccuracy, *i.e.* what is the "true state" of the object in a frame, assuming the object-originated candidate has been identified. The TAD type of approach is suitable for tracking small and fast moving objects with simple representations. Typical examples can be found in aerospace/defence applications, *e.g.* [63, 2, 55, 64, 3, 71, 38, 43, 4], where scans of radar signal are equivalent to frames in a video, and an object candidate is simply a point in a

scan with no resolvable features.

The TBD approach, on the other hand, is suitable for tracking large and slowly moving objects with complex representations. In a TBD approach, hypotheses about the object's position in a state space are generated, and then evaluated using the image. Depending on the features used for evaluating the hypotheses, this type of approach can be further categorised into contour-based tracking, colour-based tracking, *etc.* The TBD approach is often used together with a particle filter, since a particle filter provides a nice probabilistic framework for combining the hypotheses. Example applications of the TBD approach include people tracking, face tracking, *e.g.* [27, 28, 48, 45, 69]

A tennis ball is a small and fast moving object. A tennis ball has few features, which makes its representation simple. From the above discussion, the TAD approach is more suitable for tracking a tennis ball. One of the most important overall decisions we have made is to adopt the Track-After-Detection approach to our tennis ball tracking problem. We first generate ball candidates in each frame using image processing techniques. Each candidate is then treated as a two dimensional point in the image plane without resolvable features. We then try to solve a pure data association problem using only the positions of these candidates. In other words, we have decided to completely decouple detection and tracking.

Another decision we have made about the overall strategy is to make the tracker as generic as possible. For example, most abrupt motion changes of the ball happen around a player. It would then be beneficial to track the players, and use the player positions to help the ball tracking. However, we have decided not to use this piece of information, and rely only on the positions of the ball candidates. By doing this, we hoped our data association algorithms could serve as a generic tracker for small and fast moving objects that undergo "switching" dynamics.

1.4 Contributions

As mentioned earlier, a TAD approach to a tracking problem has to deal with two problems: data association and estimation. In fact, once the data association problem is solved, the estimation problem becomes trivial. For example, a Kalman smoother can give a Minimum Mean Square Estimate (MMSE) of the object state. Indeed, data association is the key in tennis ball tracking.

During the past three years, my work has been focusing on the data association side of the tracking problem. My contributions can be summarised as one tennis ball candidate detection algorithm and three data association algorithms [75, 76, 78, 77]:

I. The tennis ball candidate detection algorithm first extracts foreground moving blobs by using frame differencing, and then classifies the blobs into ball candidate and non-candidates. The main innovation of this work is a gradient-based feature for measuring the departure of the shape of a blob from an ellipse. This feature is used together with other features in an SVM classifier for the classification [75].

II. The first data association algorithm is based on sequential Monte-Carlo method, or better known as the particle filter in the computer vision community. The innovations of this work are: drawing particles directly from the *a posteriori* distribution of the object state to improve the sampling efficiency; and using the combination of smoothing and an explicit data association process, to get a more accurate tracking result with the effect of false candidates completely eliminated [75].

III. The second data association scheme is based on the Viterbi algorithm. The main innovations of this work are: first, using the Viterbi algorithm for searching the sequence of measurements with the maximum *a posteriori* probability; second, a modification to the Viterbi algorithm to allow successive misdetections of the object [76].

IV. The third data association algorithm is a three-layer one. At the bottom layer, candidate level association, the innovation is an extension to the RANdom SAmple Consensus (RANSAC) algorithm, which results in significant efficiency improvement over the baseline RANSAC algorithm. At the middle layer, tracklet level association, my contribution is to formulate the association problem as a shortest path problem in a graph with tracklets — small segments of trajectories grown from candidates — as its nodes; by exploiting a special topological property of the graph, we have also developed a more efficient all-pairs shortest path (APSP) algorithm than the general-purpose ones. Finally, at the top layer, path level analysis, the innovation is to use a path reduction (PR) algorithm to reduce the set of all-pairs shortest paths to an best set of compatible paths (BSCP), which corresponds to the set of ball trajectories [78, 77].

1.5 Overview of this Thesis

The rest of this thesis is organised as follows. Chapter 2 briefly introduces our tennis video annotation system. Tennis ball tracking is one module of this system. Chapter 3 reviews existing techniques related to this thesis. In Chapter 4, we present our ball candidate detection algorithm. This provides input data for the data association. From Chapter 5 to Chapter 7, we present the three data association algorithms we have developed, one chapter for each algorithm. Experimental results of the algorithms are presented in the corresponding chapter. We compare the three data association algorithms in Chapter 8, both mutually and with other algorithms. Finally, a summary and plans for future work are given in Chapter 9.

Chapter 2

The Tennis Video Annotation System

As previously mentioned, in the past three years, my colleagues and I have been building an automatic tennis video annotation system. The motivation of the work presented in this thesis was to provide a tennis ball tracking module for this system. Before moving on to describing the ball tracking algorithms, we first very briefly introduce our tennis annotation system.

Fig. 2 is a block diagram of the system. It should be noted that Fig. 2 is a simplified version of the system diagram. Each block in it may consist of several "modules" of the system. A detailed system diagram can be found in Fig. 2, where each block represents one module of the system. It should also be noted that my contribution to the system can be summarised as the "ball tracking" block in Fig. 2. The development and implementation of the algorithms in other blocks, as well as a memory architecture that enables the modules to communicate with each other, and a graphical interface for the system, are the work of my colleagues.

Pre-Processing Since the tennis videos used in our experiments were recorded with interlaced cameras, in the "pre-processing" block in Fig. 2, image frames are first de-interlaced into fields. Fields are used, rather than frames, in order to alleviate the effects of temporal aliasing. This is particularly important for the ball tracker. When the tennis ball is moving fast, the ball is alternately present and absent on successive frame lines, hence the need to operate on fields rather than frames. In order to keep consistent with the terminology used in other papers, we have used, and will continue to use, the word "frame" to refer to "field" in the rest of this paper.

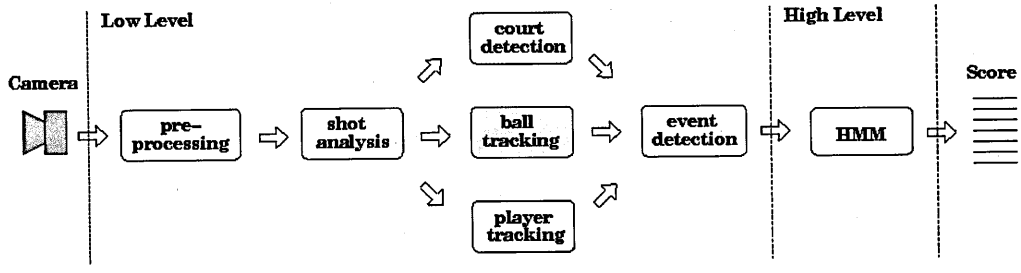


Figure 2.1: A simplified diagram of the tennis video analysis system.

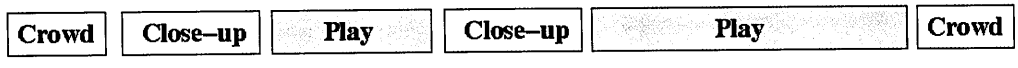
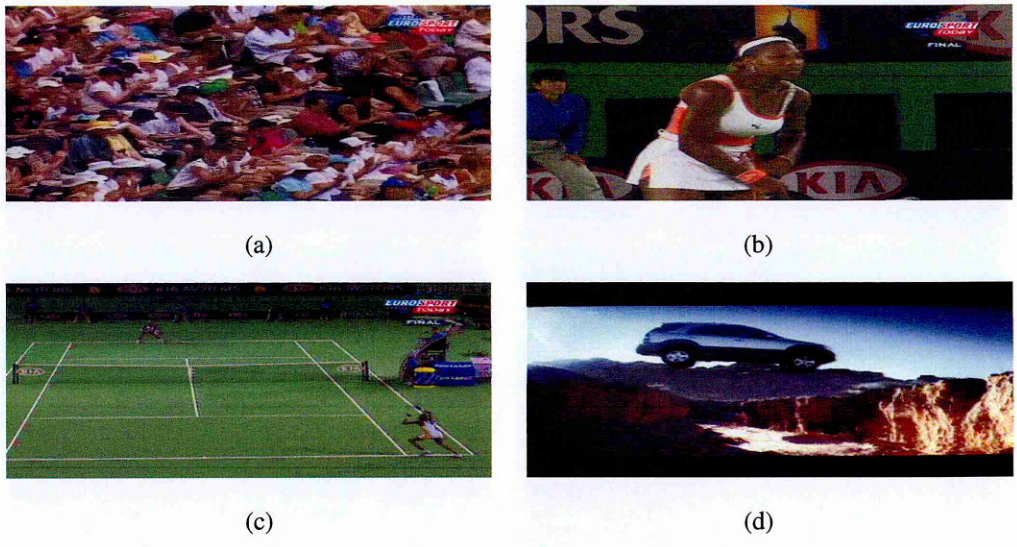


Figure 2.2: An illustrative example of the composition of a tennis video. The length of each shot is proportional to the width of the corresponding block in the figure.

After de-interlacing, the geometric distortion of camera lens is corrected. The camera position on the court is assumed to be fixed, and the global transformation between frames is assumed to be a homography [26]. The homography is found by: tracking corners through the sequence; applying RANSAC to the corners to find a robust estimate of the homography, and finally applying a Levenberg-Marquardt optimiser to improve the homography.

Shot Analysis A broadcast tennis video is composed of shots, such as play, close-up, crowd, commercial. An illustrative example of the composition of a tennis video is shown in Fig. 2. Example frames from different types of shots can be found in Fig. 2. In the "shot analysis" block of Fig. 2, shot boundaries are detected using colour histogram intersection between adjacent frames; shots are then classified into appropriate types using a combination of colour histogram mode and corner point continuity.

Court Detection, Ball Tracking, and Player Tracking For a play shot, the tennis court is detected through a combination of an edge detector and Hough transform. The players are tracked using a particle filter, and player actions are detected (see [60, 46] for details). The ball trajectories are also tracked, using techniques that will be described in this thesis.



(a) (b)

(c) (d)

Figure 2.3: Shot types. (a) crowd. (b) close-up. (c) play. (d) commercial.

Event Detection By examining the tennis ball trajectories, motion discontinuity points are detected. These points are combined with player positions, player actions and court lines in the "event detection" module, to generate key events such as hit, bounce and net.

High-Level Reasoning with HMM Finally, the generated key events are sent to a high level module, where the tennis rules are incorporated into a Hidden Markov Model (HMM). The HMM is used as a reasoning tool to generate the annotation, *i.e.* outcome of play, point awarded, *etc.* (See [35] for a details.)

Fig. 2 is a detailed diagram of our tennis annotation system. As we can see in this diagram, the system is composed of 17 modules. We have implemented a memory architecture for the modules to communicate with each other. See [34] for a detailed description of the memory architecture.

We have also implemented a graphical interface for the system (see Fig. 2). As a game progresses, detected low level features, such as player positions, ball trajectories, key events, are displayed in the main windows. The generated high level annotation, such as the score, is displayed in the left panel.

In this chapter, we very briefly introduced the automatic tennis video annotation system my

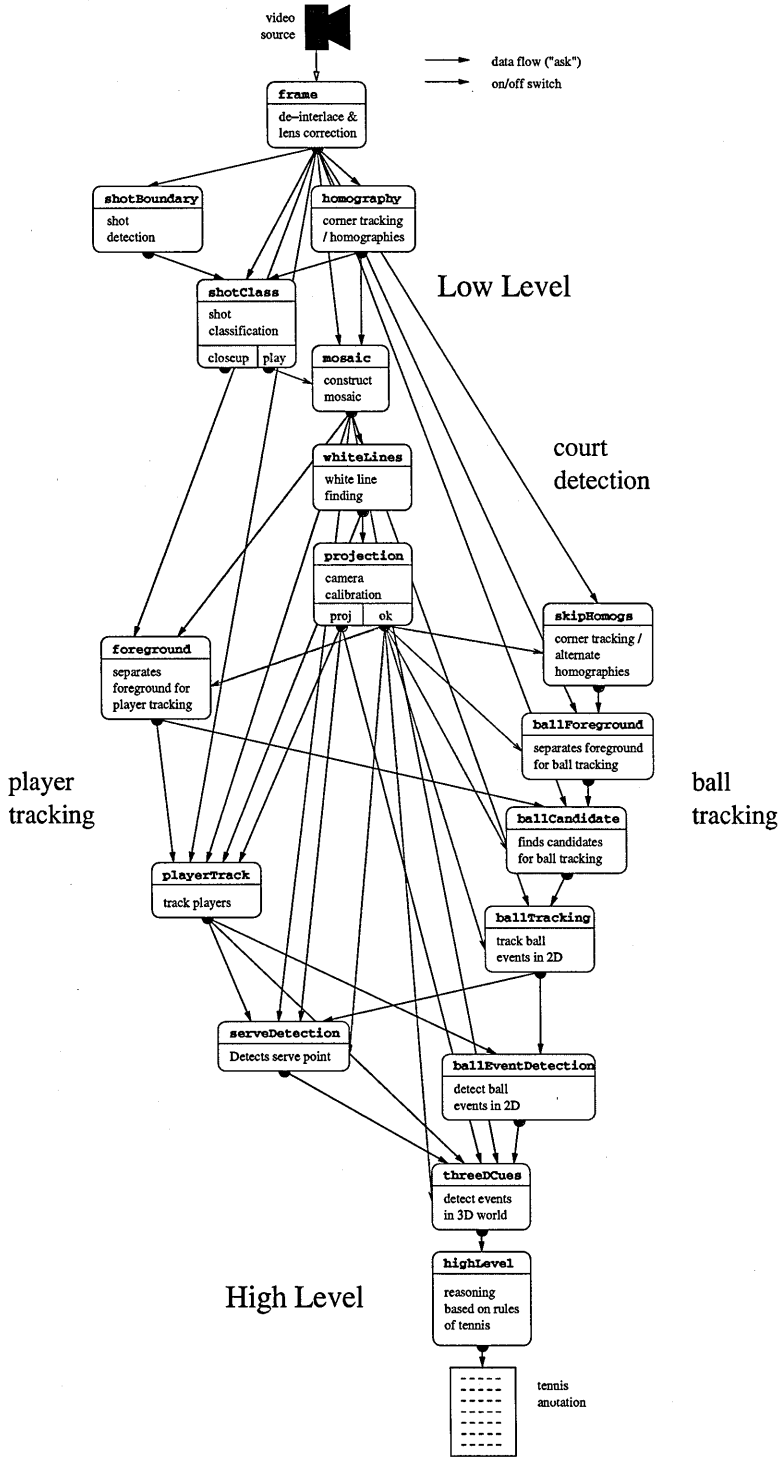


Figure 2.4: A detailed diagram of the tennis video analysis system.

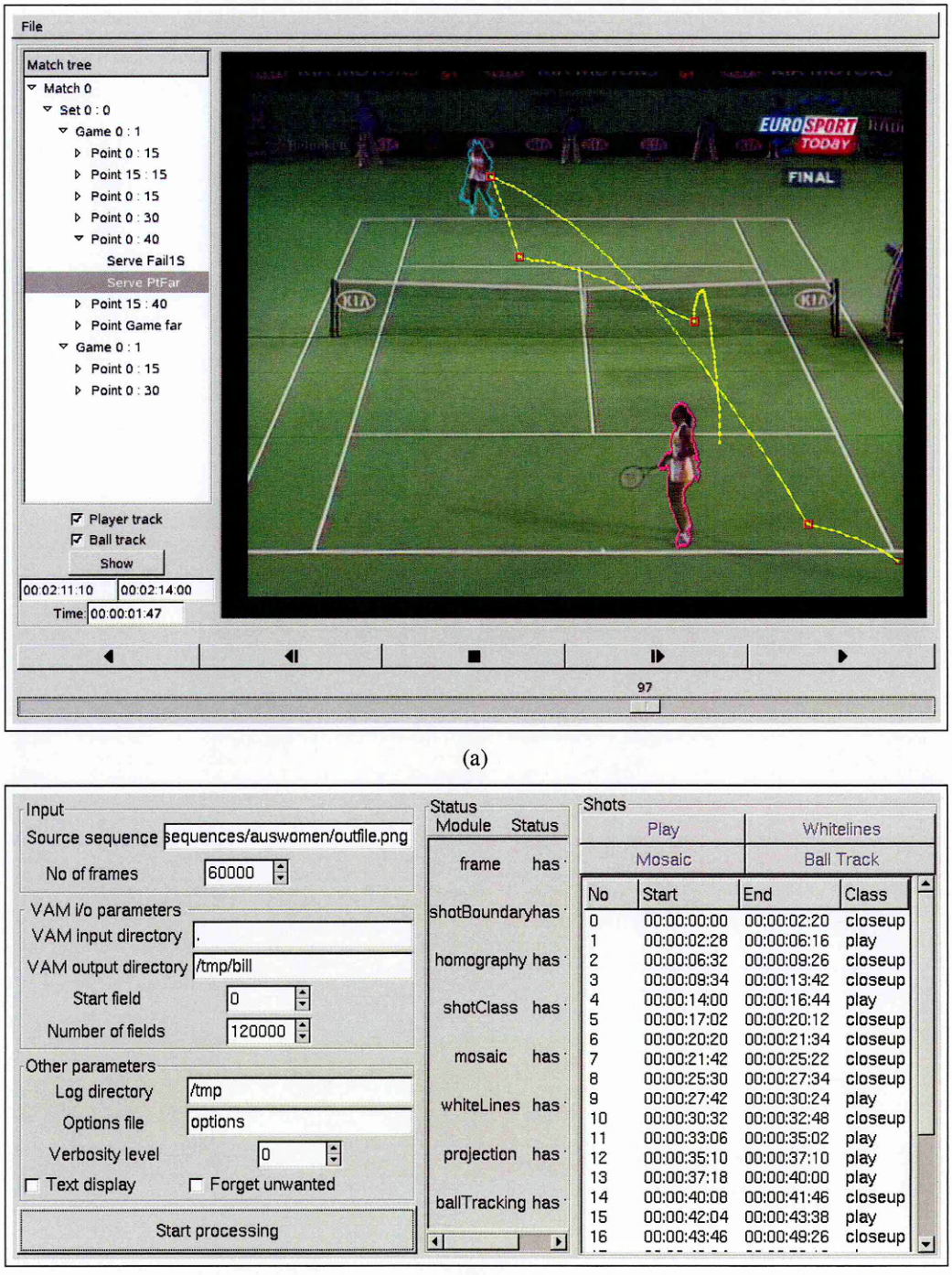


Figure 2.5: The GUI of the tennis video annotation system.

colleagues and I have been developing, including its input, its output, and the function of its main sub-systems. The tennis ball tracking algorithms proposed in this thesis were developed to provide a tennis ball tracking module for this annotation system.

Chapter 3

Literature Review

In this chapter, we give a brief review of the techniques related to this thesis, including those for data association and estimation, the sequential Monte-Carlo method, the RANSAC algorithm, and some existing tennis ball tracking algorithms.

3.1 Data Association and Estimation

As discussed in Section 1.3, a Track-After-Detection (TAD) approach to the tracking problem involves two elements: data association and estimation; and data association is the key. Estimation and data association have been intensively studied over the past several decades, especially inside the automatic control and aerospace/defence communities [2, 3, 64, 58, 55, 63, 12, 54, 40, 24, 74, 73, 4, 42, 43, 38, 53]. Most existing techniques formulate the problem as a sequential estimation problem, and data association is dealt with implicitly. A dynamic system with two models, a system model and a measurement model, is defined. The system model is used to describe how the system evolves with time:

$$\mathbf{x}_k = f_{k-1}(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{3.1}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ is the state vector of the system at discrete time k, which usually consists of the object's kinematic state, such as position, velocity, and acceleration; $\mathbf{v}_{k-1}$ is the process noise at time $k-1$; and f_{k-1} is a known, possibly nonlinear function of $\mathbf{v}_{k-1}$ and $\mathbf{x}_{k-1}$.

Let us ignore data association ambiguity for the time being. That is, let us assume there is **at most** one measurement at each time step, and that measurement is object-originated. The measurement model is in principle also a nonlinear function:

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \mathbf{w}_k) \tag{3.2}$$

where $\mathbf{z}_k \in \mathbb{R}^{n_z}$ is the measurement at time k; $\mathbf{w}_k$ is the measurement noise at time k.

Given the two models, the objective of sequential estimation is to recursively estimate $\mathbf{x}_k$ from measurements Z^k, where Z^k is the sequence of noisy measurements made on the system up to time k. Using the Bayes' rule, the *a posteriori* density $p(\mathbf{x}_k|Z^k)$ can be obtained recursively as:

$$\begin{aligned} p(\mathbf{x}_k|Z^k) &= p(\mathbf{x}_k|\mathbf{z}_k, Z^{k-1}) \\ &= \frac{p(\mathbf{z}_k|\mathbf{x}_k, Z^{k-1}) \cdot p(\mathbf{x}_k|Z^{k-1})}{p(\mathbf{z}_k|Z^{k-1})} \end{aligned} \tag{3.3}$$

where all the three involved probabilities on the right hand side of the equation can be obtained using either the system model and the measurement model, or the estimated state in the previous step.

Since the *a posteriori* probability $p(\mathbf{x}_k|Z^k)$ contains all the information about $\mathbf{x}_k$, now we are able to estimate the "optimal" $\mathbf{x}_k$ with respect to any criterion. For example, the minimum mean-square error (MMSE) estimation is the conditional mean of $\mathbf{x}_k$:

$$\hat{\mathbf{x}}_{k|k}^{MMSE} \triangleq \int \mathbf{x}_k \cdot p(\mathbf{x}_k|Z^k) d\mathbf{x}_k \tag{3.4}$$

while the maximum *a posteriori* (MAP) estimation is the the maximum of $p(\mathbf{x}_k|Z^k)$:

$$\hat{\mathbf{x}}_{k|k}^{MAP} \triangleq \arg\max_{\mathbf{x}_k} p(\mathbf{x}_k|Z^k) \tag{3.5}$$

3.2 Kalman Filter and Kalman Smoother

For the time being, let us keep the assumption that there is at most one measurement at each time step, and that that measurement is object-originated. Let us also assume f_{k-1} and h_k are linear functions, $\mathbf{v}_{k-1}$ and $\mathbf{w}_k$ are mutually independent zero-mean white Gaussian noise, and the initial density $p(\mathbf{x}_0|\mathbf{z}_0)$ is Gaussian. Under these additional assumptions, the system becomes:

$$\mathbf{x}_k = F_{k-1}\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \tag{3.6}$$

$$\mathbf{z}_k = H_k\mathbf{x}_k + \mathbf{w}_k \tag{3.7}$$

where F_{k-1} is a $n_x \times n_x$ matrix, H_k is a $n_x \times n_z$ matrix, and

$$\mathbf{v}_{k-1} \sim \mathcal{N}(\mathbf{v}_{k-1}; 0, Q_{k-1}) \tag{3.8}$$

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{w}_k; 0, R_k) \tag{3.9}$$

In this linear Gaussian system, all the involved variables are normally distributed. The mean and covariance of the *a posteriori* probability of $\mathbf{x}_k$ can be recursively estimated as:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k - H_k\hat{\mathbf{x}}_{k|k-1}) \tag{3.10}$$

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T \tag{3.11}$$

where

$$\hat{\mathbf{x}}_{k|k-1} = F_{k-1}\hat{\mathbf{x}}_{k-1|k-1} \tag{3.12}$$

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_{k-1} \tag{3.13}$$

$$\hat{\mathbf{z}}_k = H_k\hat{\mathbf{x}}_{k|k-1} \tag{3.14}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{3.15}$$

$$K_k = P_{k|k-1}H_k^T S_k^{-1} \tag{3.16}$$

These equations are collectively known as the Kalman filter [30, 70]. Note that in the case where no measurement is made at time k, (3.10) and (3.11) become:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} \tag{3.17}$$

$$P_{k|k} = P_{k|k-1} \tag{3.18}$$

The Kalman filter estimates the system state at time k using measurements up to time k. When the estimator has access to measurements "in the future", the estimate can be improved by utilising this "future" information. This is achieved through the Kalman smoother. In the Kalman smoother [3], the standard Kalman filter process is first performed for all time steps, that is, up to time K. The system state at time k, where $k < K$, is then re-estimated as:

$$\hat{\mathbf{x}}_{k|K} = \hat{\mathbf{x}}_{k|k} + B_k(\hat{\mathbf{x}}_{k+1|K} - F_{k+1}\hat{\mathbf{x}}_{k|k}) \tag{3.19}$$

$$P_{k|K} = P_{k|k} + B_k(P_{k+1|K} - P_{k+1|k})B_k^T \tag{3.20}$$

where

$$B_k = P_{k|k} F_{k+1}^T P_{k+1|k}^{-1} \tag{3.21}$$

and $\hat{\mathbf{x}}_{k|K}$ is the state vector at time k estimated in the light of measurements up to time K, and $P_{k|K}$ is its covariance.

In theory, there is an inverse relationship between the decision delay $D = K - k$ and the quality of estimation. In other words, the more future information is used, the more accurate the estimation can be.

3.3 Nearest Neighbour Standard Filter (NNSF)

When multiple measurements are made in a frame, ambiguity arises of which measurements have originated from the object being tracked. Now we are facing a data association problem. Various data association techniques have been developed to resolve this ambiguity. Among them the Nearest Neighbour Standard Filter (NNSF) [2] is the simplest one.

Before moving on to NNSF, let use first introduce the concept of gating. According to (3.14) and (3.15), the predicted value of the measurement vector is normally distributed, with a mean $\hat{\mathbf{z}}_k$ and a covariance matrix S_k. The validation region is defined as the minimum ellipse (or ellipsoid/hyper-ellipsoid) region that contains a given probability mass. Only measurements that fall into this validation region are retained for further consideration. Measurements that fall outside this region are discarded. This process is called "gating".

In NNSF, gating is performed first. The measurement that is inside the validation region and closest to the predicted value $\hat{\mathbf{z}}_k$ in (3.14) is used to update the Kalman filter (3.10) and (3.11). If no measurement is left after gating, the Kalman filter is updated using its degenerate form (3.17) and (3.18).

NNSF is simple to implement, and computationally very efficient. However, with a certain probability, it may update the Kalman filter with a measurement that has not originated from the object being tracked. In a highly cluttered environment, this may rapidly lead to track loss. To improve the robustness against clutter, more sophisticated data association techniques are needed.

3.4 Optimal Bayesian Filter (OBF)

In this section, we introduce the "best possible" data association/filtering scheme from the Bayesian point of view. Assume in the k^{th} frame of the sequence, m_k measurements are made. The total number of possible sequences of measurements at time k is then

$$L_k = \prod_{j=1}^{k} (m_j + 1) \tag{3.22}$$

where the extra 1 corresponds to the case that the object is not detected in frame j. Let us denote the l^{th} sequence of measurements at time k by $Z^{k,l} \triangleq \{Z^{k-1,s}, \mathbf{z}_{k,l}\}$, where $Z^{k-1,s}$ is the s^{th} sequence of measurements at time $k-1$, $\mathbf{z}_{k,l}$ is the measurement at time k that is in the l^{th} sequence of measurements. We define $\Theta^{k,l}$ to be the event that $Z^{k,l}$ is the "correct" sequence of measurements up to time k. Clearly, $\Theta^{k,l}$ can be decomposed as $\Theta^{k,l} \triangleq \{\Theta^{k-1,s}, \theta_{k,l}\}$, where $\Theta^{k-1,s}$ is the event that $Z^{k-1,s}$ is the correct sequence of measurements up to time $k-1$, and $\theta_{k,l}$ is the event that $\mathbf{z}_{k,l}$ is correct, *i.e.* it is the object-originated measurement at time k. We also define Z^k to be the set of sets of measurements up to time k, where $Z^k \triangleq \{Z^{k-1}, Z_k\}$, Z^{k-1} is the set of sets of measurements up to time $k-1$, and Z_k is the set of measurements at time k. According to Bayes' theorem,

$$\begin{aligned}
\beta^{k,l} &\triangleq P\{\Theta^{k,l}|Z^k\} \\
&= P\{\Theta^{k-1,s}, \theta_{k,l}|Z^{k-1}, Z_k\} \\
&\propto P\{Z_k|\theta_{k,l}, \Theta^{k-1,s}, Z^{k-1}\} \times P\{\theta_{k,l}, \Theta^{k-1,s}|Z^{k-1}\} \\
&= P\{Z_k|\theta_{k,l}, \Theta^{k-1,s}, Z^{k-1}\} \times P\{\theta_{k,l}|\Theta^{k-1,s}, Z^{k-1}\} \times P\{\Theta^{k-1,s}|Z^{k-1}\} \\
&= P\{Z_k|\theta_{k,l}, \Theta^{k-1,s}, Z^{k-1}\} \times P\{\theta_{k,l}|\Theta^{k-1,s}, Z^{k-1}\} \times \beta^{k-1,s}
\end{aligned} \tag{3.23}$$

(3.23) shows that $\beta^{k,l}$ can be computed recursively. Once $\beta^{k,l}$ are computed, $\mathbf{x}_k$ can be estimated as

$$\hat{\mathbf{x}}_{k|k} = \sum_{l=1}^{L_k} \hat{\mathbf{x}}_{k|k}^{l} \beta^{k,l} \tag{3.24}$$

where $\hat{\mathbf{x}}_{k|k}^{l}$ is the estimate given by the Kalman filter that is associated with the l^{th} sequence of measurements. This algorithm is known as the optimal Bayesian filter (OBF) [2].

3.5 Probabilistic Data Association (PDA)

In the OBF, the number of $\beta^{k,l}$ to be calculated grows exponentially. In other words, the optimality is achieved at a price of exponentially increasing complexity. Various techniques have been suggested to control the number of modes in each step. The Mixture Reduction (MR) algorithm reduces the number of modes by clustering [62]. The Probabilistic Data Association (PDA) [2] pushes this idea to its extreme, and keeps only one mode for each step. This is achieved by "regulating" the Gaussian mixture density to a single Gaussian density before propagating to the next step.

Suppose at time $k-1$, the *a posteriori* density $p(\mathbf{x}_{k-1}|Z^{k-1})$ has been "regulated" to a single Gaussian distribution. As a result, at time k, the propagated *a priori* density $p(\mathbf{x}_k|Z^{k-1})$ is also Gaussian. Let β^i be the probability that the measurement $\mathbf{z}_k^i$ is the object-originated measurement at time k, and β^0 be the probability that all the measurements are false positives. Similarly to (3.24), the system state $\hat{\mathbf{x}}_{k|k}$ can be estimated as:

$$\hat{\mathbf{x}}_{k|k} = \sum_{i=0}^{m_k} \hat{\mathbf{x}}_{k|k}^i \beta^i \tag{3.25}$$

where for $i = 1, \cdots, m_k$,

$$\hat{\mathbf{x}}_{k|k}^i = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k^i - H_k\hat{\mathbf{x}}_{k|k-1}) \tag{3.26}$$

and the special case is that for $i = 0$, *i.e.* if none of the measurements is correct,

$$\hat{\mathbf{x}}_{k|k}^0 = \hat{\mathbf{x}}_{k|k-1} \tag{3.27}$$

Note that the Kalman gain H_k is defined as in (3.16), and β^i is determined by the innovation term $\mathbf{z}_k^i - H_k\hat{\mathbf{x}}_{k|k-1}$, and is normalised to sum to one. Details of how to calculate β^i can be found in [2].

The idea of PDA can be extended to multiple objects, resulting in the Joint Probabilistic Data Association (JPDA) [2, 56]. Like PDA, JPDA is also sub-optimal, in the sense that it is an approximation to the multi-object version of the optimal Bayesian filter.

3.6 Probabilistic Multi-Hypothesis Tracker (PMHT)

A fundamental assumption made in the above data association techniques (NNSF, OBF, MR, PDA, JPDA) is that an object can produce at most one measurement in each frame. Under this assumption, the assignment of measurements, *i.e.* either object-originated or clutter-originated, is viewed as an unknown parameter of the problem. The likelihood of each measurement being object-originated or clutter-originated is enumerated, and the likelihoods are used to determine the association.

In [64], a novel data association approach is proposed under the name of Probabilistic Multi-Hypothesis Tracker (PMHT). Unlike conventional methods such as OBF and PDA, PMHT models the assignments of the measurements as independent random processes. As a result, the object being tracked can produce multiple measurements in each frame. The fundamental assumption made in the conventional methods is violated.

The basic PMHT algorithm runs in a batch mode. Let us assume a single-object tracking problem for the moment. The PMHT takes as input measurements made in several successive frames, and the objective is to estimate the probability of each measurement being object-originated/clutter-originated. This is very much like the problem of estimating the parameters of a mixture distribution from a set of data sampled from this distribution without mixture labels. In this analogy, the role of the components in the mixture density is played by the measurement model of the clutter and that of the object being tracked; and the label of each measurement, either object-originated or clutter-originated, is equivalent to the label of each datum in the mixture problem.

The Expectation-Maximisation (EM) algorithm [17] is an iterative algorithm for estimating the parameters of a random process with hidden (latent) variables. The mixture model problem described above is one of the most well-known applications of the EM algorithm. Given the analogy between the mixture model problem and the PMHT formulation of the object tracking problem, it is not surprising that the PMHT adopts the EM algorithm for estimating the probability of each measurement being object-originated/clutter-originated.

Detailed descriptions of the PMHT algorithm can be found in [64, 72, 14]. We present below a very brief summary of the basic PMHT algorithm:

1. Start from an initial value of the state of the object and an associated covariance matrix

for each frame in the batch;

2. In the **E-step**, compute the probability of each measurement being object-originated/clutter-originated;

3. In the **M-step**, compute a sequence of synthetic measurements using the current estimate of the probability of each measurement being object-originated/clutter-originated; then apply a Kalman smoother on the sequence of synthetic measurements to get a new set of estimates of the state;

4. Repeat the E-step and the M-step, until a stopping criterion is reached.

Since its introduction in the mid 1990's, the basic PMHT algorithm has been extended to track manoeuvring objects [71] and multiple objects [64]. Problems of the PMHT such as non-adaptivity, narcissism, and over-hospitality have been studied and solutions to these problems have been suggested [72, 20, 57]. Comparison between the PMHT and other tracking algorithms such as PDA can also be found in the literature [61, 56, 15].

3.7 Sequential Monte-Carlo Method

The sequential Monte-Carlo method, better known as the particle filter in the computer vision community, has its roots in Monte-Carlo integration. Let us consider the integration problem

$$I = \int f(x)\pi(x)dx \tag{3.28}$$

where $\pi(x)$ is the PDF of x. I can be approximated by Monte-Carlo integration as

$$I \approx \frac{1}{N}\sum_{i=1}^{N} f(s^i) \tag{3.29}$$

where $\{s^i, \frac{1}{N}\}_{i=1}^{N}$ is a set of uniformly weighted samples drawn according to $\pi(x)$.

When $\pi(x)$ is not directly available, but another density $q(x)$ (the importance function) is available, samples are drawn according to $q(x)$, and the integration can be approximated as

$$I \approx \frac{1}{N}\sum_{i=1}^{N} f(s^i)w(s^i) \tag{3.30}$$

where the weights

$$w(s^i) \propto \frac{\pi(s^i)}{q(s^i)} \tag{3.31}$$

Coming back to our sequential estimation problem, according to (3.3) we have:

$$p(\mathbf{z}_k|\mathbf{x}_k, Z^{k-1}) \propto \frac{p(\mathbf{x}_k|Z^k)}{p(\mathbf{x}_k|Z^{k-1})} \tag{3.32}$$

Compare (3.31) and (3.32) we can see that the desired integration

$$\hat{\mathbf{x}}_{k|k}^{MMSE} = \int \mathbf{x}_k \cdot p(\mathbf{x}_k|Z^k) d\mathbf{x}_k \tag{3.33}$$

in (3.4) can be approximated by first drawing a set of uniformly weighted samples (or "particles") $\{\mathbf{s}_k^i, \frac{1}{N}\}_{i=1}^N$ according to the *a priori* density $p(\mathbf{x}_k|Z^{k-1})$, and then calculating $\hat{\mathbf{x}}_{k|k}^{MMSE}$ as

$$\hat{\mathbf{x}}_{k|k}^{MMSE} \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{s}^i{}_k \cdot p(\mathbf{z}_k|\mathbf{s}_k^i, Z^{k-1}) \tag{3.34}$$

In (3.34), the role of the importance function $q(x)$ is played by the *a priori* density $p(\mathbf{x}_k|Z^{k-1})$, and that of the weight function $w(x)$ is played by the likelihood $p(\mathbf{z}_k|\mathbf{s}_k^i, Z^{k-1})$.

The above approximation process can be applied recursively, resulting in the sequential Monte-Carlo approach to the estimation problem. If a resampling process is used at the end of each recursion, the resulting algorithm is usually called the Sample-Importance-Resample (SIR) algorithm [59], or the CONditional DENSity propagATION (CONDENSATION) algorithm [27]. Starting from a set of uniformly weighted samples $\{\mathbf{s}_{k-1}^i, \frac{1}{N}\}_{i=1}^N$ at time $k-1$, a single update cycle of the CONDENSATION algorithm goes as follows:

1. **Predict** the new state by sampling $\mathbf{s}_k^i \sim p(\mathbf{x}_k|\mathbf{x}_{k-1} = \mathbf{s}_{k-1}^i)$.

2. **Weight** the new samples using the likelihood:

$$\pi_k^i = p(\mathbf{z}_k|\mathbf{s}_k^i, Z^{k-1}) \tag{3.35}$$

   then normalise so that $\sum_{i=1}^N w_k^i = 1$. The *a posteriori* density $p(\mathbf{x}_k|Z^k)$ is approximated by a set of non-uniformly weight particles, $\{\mathbf{s}_k^i, w_k^i\}_{i=1}^N$. $\hat{\mathbf{x}}_{k|k}^{MMSE}$ can then be estimated as:

$$\hat{\mathbf{x}}_{k|k}^{MMSE} \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{s}^i{}_k w_i^k \tag{3.36}$$

3. **Resample** $\mathbf{s}_k^i$. This is done by selecting N particles from the old set $\{\mathbf{s}_k^i, w_k^i\}_{i=1}^N$, in such a way that the probability of selecting a particle $\mathbf{s}_k^i$ in the old set is w_k^i. After resampling, the *a posteriori* density $p(\mathbf{x}_k|Z^k)$ is approximated by a set of uniformly weight particles, $\{\mathbf{s}_k^i, \frac{1}{N}\}_{i=1}^N$.

Due to its nature of sampling, the CONDENSATION algorithm can handle multi-modal densities automatically. It can also be used in the situation where the system is non-linear and/or non-Gaussian.

The *a priori* density $p(\mathbf{x}_k|Z^{k-1})$ is convenient to draw samples from. However, if it is a much broader distribution than the likelihood $p(\mathbf{z}_k|\mathbf{x}_k)$, using it as importance function may lead to the degeneracy phenomenon. That is, after certain steps, all but one particle will have negligible normalised weights [59]. Techniques have been developed to cope with this problem, by combining the current measurement into the importance function, to "herd" the particles to the right part of the state space [28, 52]. The resampling step also helps reduce the degeneracy effect, but at the same time it may cause "sample impoverishment" - the lack of diversity among the particles. Solutions to this problem have been suggested in [59].

3.8 The RANdom SAmple Consensus (RANSAC) Algorithm

The RANSAC (RANdom SAmple Consensus) algorithm is a robust algorithm for estimating model parameters. Since its introduction in 1981 [21], RANSAC has been successfully applied in a wide range of computer vision problems, including stereo matching, motion estimation, object recognition, *etc.* The baseline RANSAC algorithm operates as follows:

1. **Draw** a subset of M sample points randomly from the observed data, where M is the minimal number required to solve for the model parameters;
2. **Solve for** the model parameters using the drawn sample set;
3. **Evaluate** the hypothetical model using the full data set and a cost function.

The above loop is repeated until the probability that a "good" model has been found is higher than a threshold.

RANSAC is a robust algorithm in the sense of good tolerance to outliers. However, simple math shows that, as the proportion of true positives drops and as the dimension of model parameter increases, the probability of getting an "uncontaminated" sample set decreases polynomially, where the polynomial coefficient is the size of a sample set. More precisely, let I be the number of true positives (inliers), N be the number of all data, $\varepsilon \triangleq \frac{I}{N}$ be the ratio between them, and m be the size of each sample set. The probability of getting an uncontaminated sample is [10]:

$$P = \frac{\binom{I}{m}}{\binom{N}{m}} = \prod_{j=0}^{m-1} \frac{I-j}{N-j} \approx \varepsilon^m \tag{3.37}$$

The number of trials K required is set such that the probability that all of the trials are contaminated, η, is smaller than a pre-defined threshold η_0. It has been shown in [10] that

$$K \geq \frac{\log(1-P)}{\log(1-\varepsilon^m)} \tag{3.38}$$

In other words, K grows fast as ε drops and as m increases. This fast growing complexity means a theoretically solvable problem can still be computationally challenging.

In recent years, various techniques have been proposed to improve the RANSAC algorithm [65, 7, 9, 10, 8, 44, 5, 66, 37, 67]. One category of the techniques tries to draw sample sets according to some *a priori* knowledge instead of uniformly, thus improving the probability of drawing an uncontaminated sample set. In [65], a guided RANSAC algorithm is proposed for homography estimation between an image pair. The probability of each pair of features being "true" is estimated using the zero-normalised cross-correlation between the two features. The probabilities are then used to guide the sampling. Pairs that are more likely to be true are sampled more frequently. In [8], a PROgressive SAmple Consensus (PROSAC) algorithm is described. Instead of having to estimate the probability of each datum being true as in [65], PROSAC works under a weaker assumption that some quantity monotonically related to the probability is available. Samples with better quality are then drawn more frequently. In [44], an N Adjacent Points SAmple Consensus (NAPSAC) algorithm is proposed. The key idea is that given outliers possess a diffuse distribution, sampling the minimal sets based on proximity can increase the probability of selecting an all-inlier sample set. Methods that apply guided sampling recursively have also been reported in the literature. For example, in [66], a coarse-to-fine strategy is adopted to estimate epipolar geometry. RANSAC is applied at each level; the quality of each model in the coarser level is propagated into finer level, and used to guide the sampling in the finer level.

Another attempt to improve the baseline RANSAC algorithm is to try to reject a "bad" model at an early stage of the evaluation process. This is usually achieved by randomising the model evaluation step of the RANSAC algorithm [7, 37, 5]. Instead of evaluating the hypothetical model using the full data set, a randomly selected small subset of the full data set is used to pre-test the model. The model is evaluated using the full set only if it passes the pre-test.

In [9, 10], the assumption that an uncontaminated data set always leads to a "good" model is challenged. This results in a Locally Optimised RANSAC (LO-RANSAC) algorithm. The key idea of LO-RANSAC is to introduce an extra optimisation step to the RANSAC loop to refine the model using all the inliers found so far. In [67], a new cost function is suggested, resulting in a new RANSAC variant called the Maximum Likelihood Estimator RANSAC (MLESAC). Instead of maximising the number of inliers, MLESAC maximises a likelihood function. MLESAC outperforms baseline RANSAC in terms of accuracy of estimation, because the likelihood function used in MLESAC is a more accurate assessment of fit than the number of inliers. However, this is achieved at the price of more computation, since the parameters of the likelihood function have to be estimated.

3.9 Existing Tennis Ball Tracking Algorithms

Several tennis ball tracking algorithms have been reported in the literature [50, 51, 39, 47, 36, 80]. All these algorithms adopt the TAD approach: first ball candidates are generated using image processing techniques; the tracking problem is then treated as a data association and estimation problem. Some of the existing algorithms have been successfully applied in practice for enhanced broadcast [51, 47]. In [51, 47], multiple cameras are used. This allows "cross validation" of tracking results. As a result, the data association ambiguity is easier to resolve. Another benefit of using multiple cameras is that the final reconstructed ball trajectories are in the 3D real world space, instead of in the 2D image plane.

Techniques in [50, 39, 36, 80], on the other hand, rely on a single camera. In this sense, they are more closely related to the scope of this thesis. In [50, 39, 80], the focus is on ball candidate generation rather than data association. Usually foreground moving objects are extracted by frame differencing. The resulting blobs are then tested using visual features such as size, shape and

colour. Among the blobs that have passed the test, the one that is closest to a predicted position is used to update the trajectory. In other words, the data association problem is implicitly addressed with a nearest neighbour type of single hypothesis scheme. This inevitably imposes the assumption that the false candidate rate is very low, and the true ball detection rate is reasonably high.

In [36], the authors propose a data association algorithm under the name of Robust Data Association (RDA), and use RDA to track a tennis ball in monocular sequences. RDA is the state-of-the-art algorithm for tennis ball tracking in monocular sequences. It does not use the naive single hypothesis data association scheme. The key idea of RDA is to treat data association as a motion model fitting problem. First, ball candidates in each frame are detected. A sliding window containing several frames is then moved over the sequence. A RANSAC-like algorithm is used to find the motion model that is best at explaining the candidates inside the window. An estimate of the ball position in one frame, *e.g.* the middle frame in the sliding window, is then given by this model. As the sliding window moves, eventually ball positions in all frames are estimated.

3.10 Conclusions

In this chapter, we briefly reviewed some existing techniques related to this thesis. As we pointed out in Chapter 1, resolving data association ambiguity is the key to the tennis ball tracking problem. On the one hand, data association has been intensively studied inside the automatic control and aerospace/defence communities, and many "classical" techniques have been proposed. On the other hand, most existing tennis ball tracking algorithms have been developed inside the image processing and computer vision communities. They focus on the image processing side of the problem, *i.e.* they assume a very high quality of ball candidate detection, as a result, the data association problem is trivialised.

In this thesis, we try to bridge the gap between the classical data association techniques and the tennis ball tracking problem. Offline tennis ball tracking in monocular sequences differs from typical automatic control and aerospace/defence applications in that: first, it poses its own challenges, such as abrupt change of motion direction, to the classical techniques; second, it also

relaxes some of the requirements in classical applications, for example, ball tracking does not have to be an online process, meaning that "future" information can be used to help the estimation/association. In this thesis, based on the classical data association techniques, we develop two algorithms that can both cope with the new challenges, and take advantage of the relaxations. We also look closely at the Robust Data Association (RDA) algorithm. We identify the weaknesses of the RDA, and propose remedies to them, which result in a third data association algorithm.

Chapter 4

Ball Candidate Detection

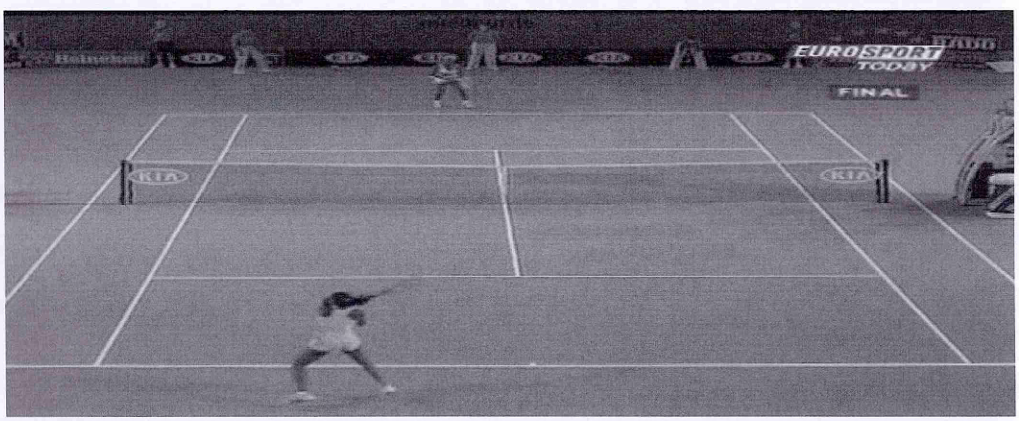
4.1 Abstract

As mentioned in Section 1.3, we have decided to adopt the TAD approach to the tracking problem. All of our three proposed data association algorithms are driven by a ball candidate detection module. In this chapter, we present a ball candidate detection algorithm, which provides input data for the data association algorithms. For each frame in a play shot, foreground moving blobs are detected using frame differencing. The blobs are then classified into ball candidates and non-candidates. The main innovation of this work is a gradient-based feature for measuring the departure of the shape of a blob from an ellipse. This feature is used together with other features in an SVM for the classification.

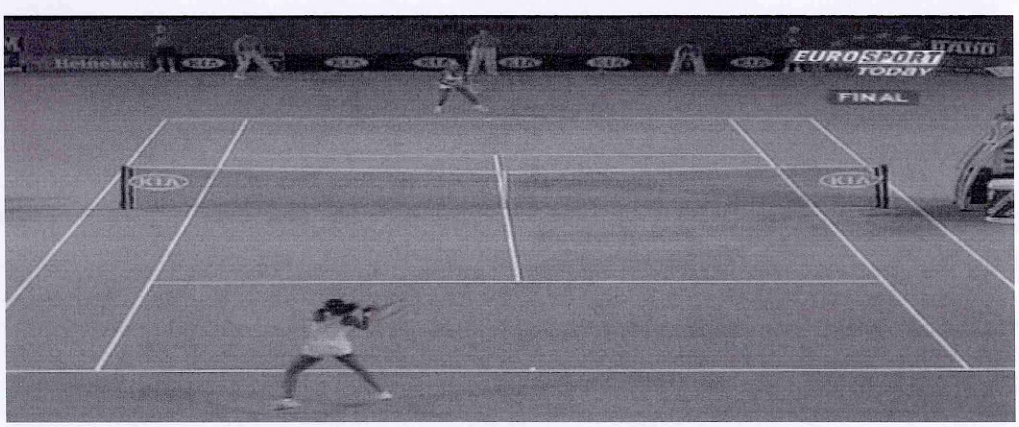
4.2 Motion Segmentation

Ball tracking is performed only on play shots. From now on, we refer to a play shot as a "sequence". Assume that a sequence is composed of K frames, and that the frames are numbered from 1 to K. Also assume that homographies between frames have been calculated by tracking corresponding corners, and global motion between frames has been compensated using the homographies. Pixel-wise temporal differencing is then adopted as a means of motion detection.

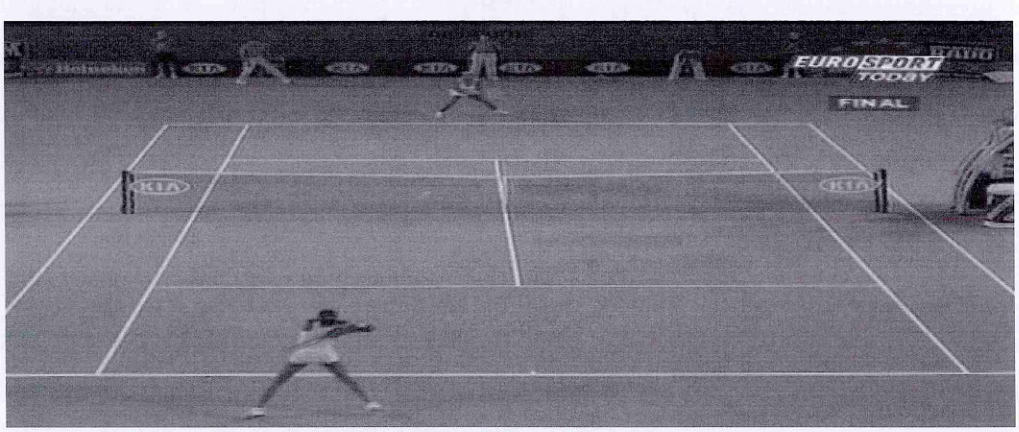
We first convert the frames from the RGB space to an intensity space, where the intensities range



(a)

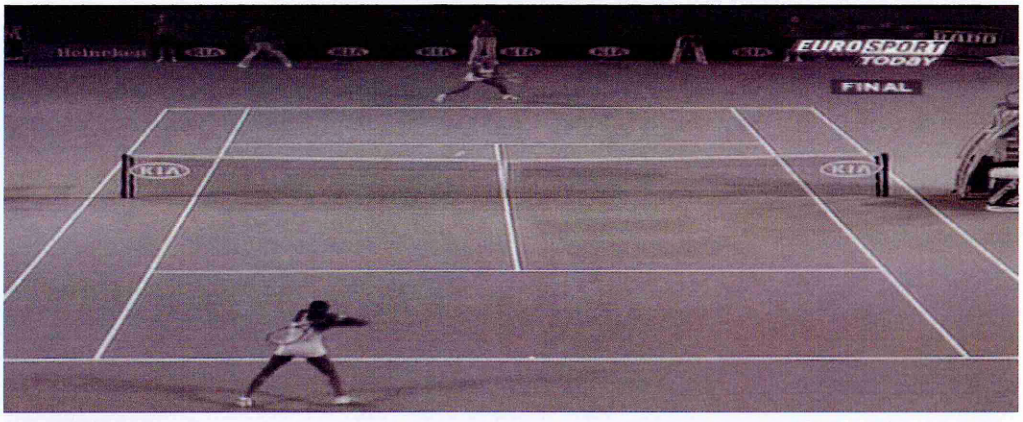


(b)

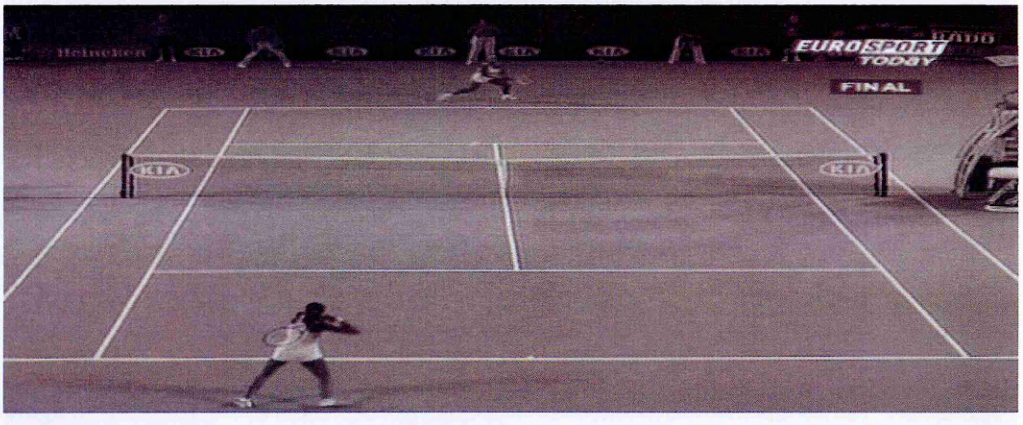


(c)

Figure 4.1: Motion segmentation - part 1. From (a) to (c): frame 193, 195 and 199 after being converted to intensity chanel, respectively.



(a)



(b)

(c)

Figure 4.2: Motion segmentation - part 2. (a) and (b): frame203 and 205 after being converted to intensity chanel, respectively. (c): the result of motion segmentation of frame 199.

from 0 to 255. A 5×5 Gaussian smoother is then applied to each frame to reduce noise. Let f_k be the k^{th} frame in the sequence after the conversion and smoothing. Also let f_l be a neighbouring frame after the conversion and smoothing, and f_l is already motion compensated with respect to f_k, using the homography between them. Under the assumption that a pixel inside the ball region is brighter than a pixel that belongs to background, for a pixel $p_k(x, y)$ located at row x and column y in f_k and with intensity $I_k(x, y)$, a boolean variable is defined as

$$FG_{k,l}(x, y) \triangleq \{I_k(x, y) - I_l(x, y) > i_{th}\} \tag{4.1}$$

where i_{th} is a threshold. A pixel $p_k(x, y)$ is classified as foreground pixel if

$$FG_k(x, y) = true \tag{4.2}$$

where $FG_k(x, y)$ is defined as

$$FG_k(x, y) \triangleq \bigwedge_{\forall l \in L_k} \mathrm{FG}_{k,l}(x, y) \tag{4.3}$$

where $\wedge$ is the logical AND operation, and L_k is a collection of several neighbouring frames. In our implementation,

$$L_k = \{k - 6, k - 4, k + 4, k + 6\} \tag{4.4}$$

Note that only frames that are even number of frames away from f_k are used for motion segmentation. This is because de-interlacing introduces a half-line difference between even numbered frames and odd numbered frames, and this will reduce the accuracy of the homography computation. Also note that f_{k-2} and f_{k+2} are not used, to prevent the ball being excluded when it is travelling at very low velocity.

4.3 Foreground Blob Classification

The output of motion segmentation is a binary mask image, as shown in Fig. 4.2 (c), where a "true" (white pixel) corresponds to a foreground moving pixel, and a "false" (black pixel) corresponds to a background pixel.

The foreground pixels are clustered into blobs according to their connectedness using a 4-connected neighbourhood system. A blob may have originated the true tennis ball; it may also

have originated from part of a player, a racket, or a advertising board, due to various inaccuracies. A method for foreground blob classification is required. In [50], the authors suggest that the tennis ball has a standardised yellow colour, which can be used as an important cue for the classification. However, in some off-air material, since the colour bandwidth is low, and the tennis ball has a very small size, its colour can be strongly affected by the colour of the background it is travelling on (see Fig.1.2 (b)). Moreover, off-air material may be subject to artifacts introduced by analogue encoding, for example, the PAL cross-colour effect, which appears as multicoloured noise "floating" in the image (see Fig.1.2 (c)). This suggests that colour information by itself is not sufficient for the classification. We propose a multi-cue foreground blob classification method as follows.

For each foreground blob, a small surrounding area is included, and the enlarged blob is interpolated, using bi-linear interpolation. The corresponding binary mask is also interpolated. The rule for this binary interpolation is defined as:

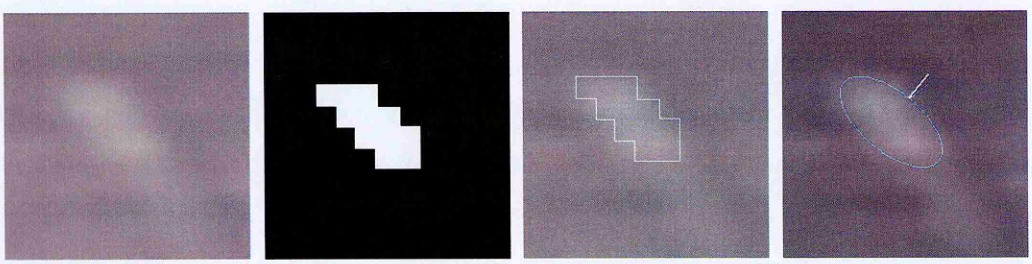
$$B(x',y') \triangleq B(x,y) \wedge B(x+1,y) \wedge B(x,y+1) \wedge B(x+1,y+1) \tag{4.5}$$

where $B(x',y')$ is the value of the interpolated pixel, and $B(x,y)$, $B(x+1,y)$, $B(x,y+1)$, $B(x+1,y+1)$ are values of the four pixels in the binary mask used for the interpolation. The edge pixels of the interpolated binary image are then extracted, and used as the edge of the blob in the interpolated intensity image. An ellipse is fitted to the edge pixels, using a least squares ellipse fitting algorithm [25]. M points are sampled along the ellipse. For each point, the normal direction is found, and the gradient is calculated using a 3×3 Sobel mask.

Fig. 4.3 shows the major steps involved in this process. Once the normal directions and gradient directions at the sample points are found, the mean of the absolute angle difference of the normal directions and the gradient directions at all sample points can be used for foreground blob classification. We define

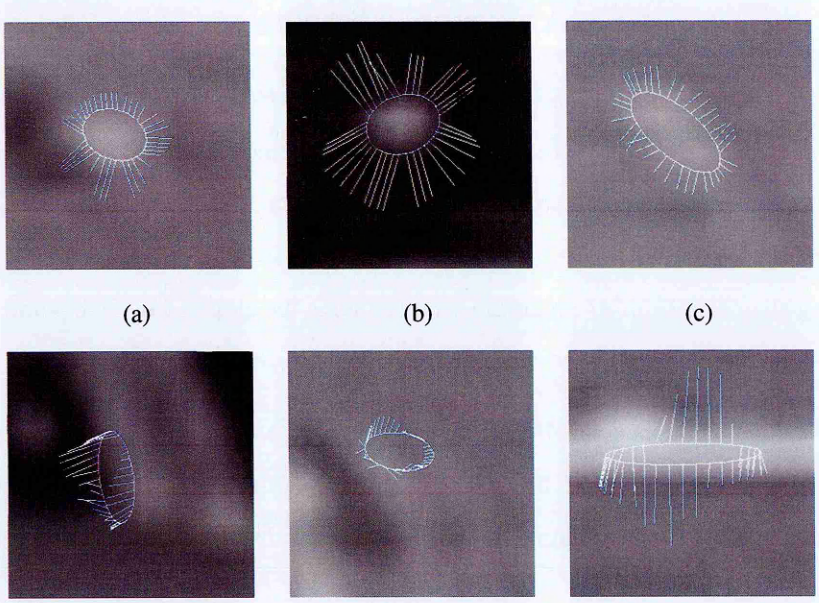
$$\bar{\alpha} \triangleq \frac{1}{M} \sum_{i=1}^{M} \alpha_i \tag{4.6}$$

where α_i is the absolute angle difference in radians at the i^{th} sample point. It takes a value ranging from 0 to π. Using the assumption that a ball-originated blob is a local maximum in the intensity image and is approximately elliptical (circular when the ball is stationary), a blob with a smaller $\bar{\alpha}$ is more likely to be ball-originated (see Fig. 4.3).



(a) (b) (c) (d)

Figure 4.3: An important feature for blob classification: absolute angle difference between gradient orientation and normal orientation. (a) interpolated intensity image. (b) interpolated binary image. (c) interpolated intensity image with blob edge. (d) fitted ellipse with the normal (thick line) and Sobel gradient (thin line) at a sample point.



(a) (b) (c)

(d) (e) (f)

Figure 4.4: Using $\bar{\alpha}$ for blob classification. Top row: gradient vectors of ball-originated blobs. Bottom row: gradient vectors of clutter-originated blobs. (a) $\bar{\alpha} = 0.24$, (b) $\bar{\alpha} = 0.27$, (c) $\bar{\alpha} = 0.31$, (d) $\bar{\alpha} = 1.82$, (e) $\bar{\alpha} = 1.93$, (f) $\bar{\alpha} = 1.36$.

$\bar{\alpha}$ is used as one dimension of an 8-dimensional feature vector. Other dimensions include the mean of the pixels inside the blob in HSV channels $\bar{h}$, $\bar{s}$, and $\bar{v}$, the size of the blob (major axis a and minor axis b of the fitted ellipse), and the position of the blob centre (row x and column y). The feature vector is then

$$\mathbf{f} = \{\bar{\alpha}, \bar{h}, \bar{s}, \bar{v}, a, b, x, y\} \tag{4.7}$$

We used foreground blobs in 5 sequences from the 2003 Australian Open women's final as training data. We manually labelled 1000 ball-originated blobs sequences as positive examples and 1000 clutter-originated blobs as negative examples. An SVM classifier [68] with a Radial Basis Function (RBF) kernel is trained using the 2000 examples.

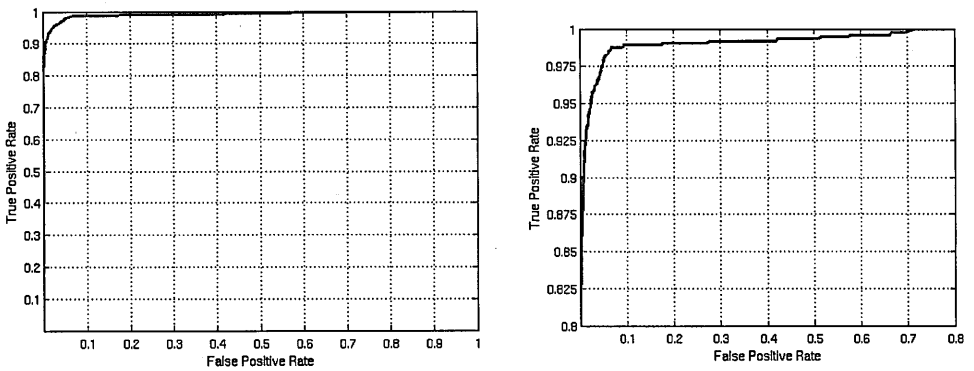
4.4 Experiments

Experimental Data We used another 70 sequences from the 2003 Australian Open women's final game as the test set for our experiments. Each sequence contains one play starting from a serve, in other words, there is at most one ball in each frame. In total there are 25,158 frames in the 70 sequences. Since ball candidate detection is a two-step process: first motion segmentation and then foreground blob classification, we present below the experimental results of the two steps separately.

Motion Segmentation In the first step, motion segmentation, foreground blobs are extracted by frame differencing. In total there are 22178 frames with a ball in the field of view (FOV). However, due to occlusion and motion blur (see Fig.1.2), not all balls in the FOV are extracted as a foreground blob. In total there are 354,257 foreground blobs extracted, and among them 20,647 are ball-originated. The probability of a ball in the FOV being extracted as a foreground blob is then:

$$r_b = \frac{\text{number of balls in the FOV being extracted as blobs}}{\text{number of balls in the FOV}} = \frac{20,647}{22,178} \approx 0.931 \tag{4.8}$$

Foreground Blob Classification In the second step, foreground blob classification, the extracted foreground blobs are classified into ball candidates and non-candidates. Among the 354,257 foreground blobs, 20,647 are ball-originated. We manually labelled the 20,647 blobs as



(a) (b)

Figure 4.5: Receiver Operating Characteristic (ROC) curve of foreground blob classification using the SVM classifier. (a) The original ROC curve. (b) A zoomed-in version of the ROC curve.

ball-originated, and the other 333,610 blobs are labelled as clutter-originated. The trained SVM was then applied to classify these blobs. This is a standard two-class classification problem. We measure the classifier's performance using true positive rate and false positive rate.

By moving the decision boundary of the SVM classifier, a trade-off can be made between the true and false positive rates. The true positive rate r_{tp} is defined as

$$\begin{aligned} r_{tp} &= \frac{\text{number of true positives}}{\text{number of positive instances}} \\ &= \frac{\text{number of ball-originated blobs being classified as ball candidates}}{\text{number of ball-originated blobs}} \end{aligned} \tag{4.9}$$

The false positive rate r_{fp} is defined as

$$\begin{aligned} r_{fp} &= \frac{\text{number of false positives}}{\text{number of negative instances}} \\ &= \frac{\text{number of clutter-originated blobs being classified as ball candidates}}{\text{number of clutter-originated blobs}} \end{aligned} \tag{4.10}$$

The Receiver Operating Characteristic (ROC) curve, which plots r_{tp} against r_{fp}, is shown in Fig. 4.4.

Several sample points on the ROC curve are shown in Table 4.4. We can clearly see in Table 4.4 the trade-off between r_{tp} and r_{fp}. More precisely, higher r_{tp} means a ball-originated blob is more likely to be classified as a candidate, but this is achieved at the cost of introducing more false positives, *i.e.* classifying more clutter-originated blobs as candidates.

Table 4.1: Several sample points on the ROC curve

SVM boundary	-4	-3	-2	-1	0	1	2
r_{tp}	**1**	**1**	**0.992**	**0.970**	**0.887**	**0.617**	**0.048**
r_{fp}	**0.987**	**0.810**	**0.319**	**0.044**	**0.007**	**0**	**0**
r_d	0.931	0.931	0.924	0.903	0.826	0.574	0.045
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0	0

Having defined r_b and r_{tp}, the probability of a ball in the FOV being detected as a candidate, or the "ball detection rate", is then

$$r_d = r_{tp} \times r_b \tag{4.11}$$

The ball detection rate r_d and the average number of false candidates in each frame $\bar{N}$, which is determined by r_{fp}, are also shown in Table 4.4. According to Table 4.4, setting the SVM boundary -1 seems a good choice. This gives an r_d of $0.970 \times 0.931 = 0.903$, and an $\bar{N}$ of only 0.6. Since the positions of the candidates are used as the input for the data association process that follows, r_d and $\bar{N}$ have significant impact on the performance of a data association algorithm, as we will see later in this thesis. It should be noted that since the camera has pan, tile and zoom (PTZ), candidate positions in different frames are first projected into a common coordinate system using the homographies, before they are used for data association.

Processing Speed The speed of the proposed ball candidate detection algorithm was also measured. On average, it takes 0.13 seconds to perform motion segmentation on each frame. This includes the time spent on Gaussian smoothing, frame differencing, and pixel clustering. On average, it takes 0.29 seconds to perform blob classification on each frame, where there are on average 14.1 blobs. In total, it takes 0.42 seconds to perform ball candidate detection on each frame, which makes the processing rate 2.4 frames per second. The time spent on training the SVM classifier is approximately 1.63 seconds. Time was measured on a single thread of an Intel Xeon 3.0G computer running Linux, and overhead of disk I/O was included. All algorithms presented in this thesis are implemented in C++ with the Recognition and Vision Library

(RAVL) [1].

4.5 Conclusions

In this chapter, we presented a ball candidate generation algorithm. The algorithm takes as input frames in a sequence. It extracts foreground moving objects by taking the difference of nearby frames and thresholding the difference. The resulting foreground pixels are clustered into blobs. The blobs are classified into ball candidates and non-candidates using multiple visual cues and an SVM classifier. Experiments show that good detection rate and low false candidate rate can be achieved with the proposed method.

Chapter 5

Data Association with Sequential Monte-Carlo Method

5.1 Abstract

In this chapter, we present a data association algorithm based on the sequential Monte-Carlo method. The first innovation of this work is that we derive a closed form expression of the *a posteriori* probability of the state vector being estimated, and draw samples directly from this *a posteriori* density. As a result, an improved sampling efficiency is achieved. We also use a second dynamic model to deal with abrupt motion changes of the ball. Our specific application of automatic annotation requires the reconstructed ball trajectories to be very accurate. After the filtering pass, smoothing is applied to improve the accuracy of the estimation. Finally, as the second innovation of this work, an explicit data association process is performed, to completely eliminate the effect of false candidates.

5.2 Problem Formulation

Tennis ball tracking is formulated as a sequential estimation problem, as defined in Section 3.1. We assume that the system is linear and time invariant, the system model (3.1) and measurement

model (3.2) become:

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + \mathbf{v} \tag{5.1}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{w} \tag{5.2}$$

respectively, where $\mathbf{x}_k$ is the system state, $\mathbf{z}_k$ is the measurement, and $\mathbf{v}$, $\mathbf{w}$ are process noise and measurement noise, respectively. We assume the sequential estimation process has been initialised. The objective is to estimate $\mathbf{x}_k$ recursively, using the noise corrupted measurement $\mathbf{z}_k$ and the estimate of $\mathbf{x}_{k-1}$. In our ball tracking problem, the position (x_k, y_k), the velocity $(\dot{x}_k, \dot{y}_k)$ and the acceleration $(\ddot{x}_k, \ddot{y}_k)$ of the ball in the image plane are modelled in the state vector:

$$\mathbf{x}_k = (x_k, y_k, \dot{x}_k, \dot{y}_k, \ddot{x}_k, \ddot{y}_k)^T \tag{5.3}$$

Assume the acceleration of the ball is constant in two successive frames. According to Newton's laws of motion,

$$\mathbf{s} = \mathbf{s_0} + \mathbf{v_0}t + \frac{1}{2}\mathbf{a}t^2 \tag{5.4}$$

where $\mathbf{s}$ is the position of an object at time t, $\mathbf{s_0}$ is the position of the object at time 0, $\mathbf{v_0}$ is the velocity of the object at time 0, and $\mathbf{a}$ is the constant acceleration. The state transition matrix F is then:

$$F = \begin{pmatrix} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.5}$$

Since the velocity and acceleration of the ball are not directly observable, the measurement matrix H is:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.6}$$

The process noise $\mathbf{v}$ and measurement noise $\mathbf{w}$ are assumed to be zero-mean white Gaussian random processes, with covariance matrices Q and R, respectively.

Using the assumptions made above, we have:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \frac{1}{(2\pi)^2|Q|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{x}_k - F\mathbf{x}_{k-1})^T Q^{-1}(\mathbf{x}_k - F\mathbf{x}_{k-1})] \tag{5.7}$$

Assume the number of clutter-originated measurements is Poisson distributed. The total measurement density can be proved to be, up to proportionality [27],

$$p(\mathbf{z}_k|\mathbf{x}_k) \propto \beta + \sum_{j=1}^{m_k} \frac{1}{2\pi|R|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{z}_k^j - H\mathbf{x}_k)^T R^{-1}(\mathbf{z}_k^j - H\mathbf{x}_k)] \tag{5.8}$$

where $\mathbf{z}_k^j$ is the j^{th} measurement (position of candidate), m_k is the number of candidates detected at time k, and β is a constant corresponding to the measurement density caused by clutter.

According to (5.7), (5.8) and (3.3), the *a posteriori* density $p(\mathbf{x}_k|Z^k)$ can be derived to be a Gaussian mixture with ever growing number of components. The optimal estimator of the state, the Optimal Bayesian Filter (OBF), was introduced in Section 3.1. In the OBF, the decomposition of densities with respect to all the measurements is performed at each time step, leading to an exponentially increasing number of modes in the Gaussian mixture, and hence an exponentially increasing computational complexity. On the other hand, Mixture Reduction (MR) and Probabilistic Data Association (PDA) have much lower complexity, but at the price of loss of optimality.

In this chapter, we propose a solution to this sequential estimation problem based on sequential Monte-Carlo method, or better known as the particle filter in the computer vision community. With the proposed algorithm, the computational complexity is limited without losing Bayesian optimality. Moreover, the degeneracy problem of particle filtering is also dealt with automatically.

5.3 Particle Filtering with an Improved Sampling Efficiency

Assume at time $k-1$ the *a posteriori* density of $\mathbf{x}_{k-1}$ is represented by a set of particles with uniform weights $\{\mathbf{s}_{k-1}^i, \frac{1}{N}\}_{i=1}^N$, *i.e.* ,

$$p(\mathbf{x}_{k-1}|Z^{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \delta(\mathbf{x}_{k-1} - \mathbf{s}_{k-1}^i) \tag{5.9}$$

where N is the number of particles, $\delta(\cdot)$ is the Dirac delta function, *i.e.*

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \tag{5.10}$$

and Z^{k-1} is the measurement history up to time $k-1$. Using (5.9) and (5.7), the *a priori* density of $\mathbf{x}_k$ can be obtained as:

$$p(\mathbf{x}_k|Z^{k-1}) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi)^2|Q|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{x}_k - F\mathbf{s}_{k-1}^i)^T Q^{-1}(\mathbf{x}_k - F\mathbf{s}_{k-1}^i)] \tag{5.11}$$

According to Bayes' theorem and using the result of (5.11) and (5.8), after some rearrangement, we get the *a posteriori* density at time k as:

$$p(\mathbf{x}_k|Z^k) \propto \sum_{i=1}^{N}\sum_{j=0}^{m_k} w_k^{i,j} \frac{1}{(2\pi)^2|C_k^{i,j}|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{x}_k - \mathbf{m}_k^{i,j})^T {C_k^{i,j}}^{-1}(\mathbf{x}_k - \mathbf{m}_k^{i,j})] \tag{5.12}$$

where

$$w_k^{i,0} = \beta, \quad C_k^{i,0} = Q, \quad \mathbf{m}_k^{i,0} = F\mathbf{s}_{k-1}^i \tag{5.13}$$

for $j \neq 0$

$$\begin{aligned} w_k^{i,j} &= \frac{1}{2\pi|R+HQH^T|^{\frac{1}{2}}} \exp[-\tfrac{1}{2}(\mathbf{z}_k^j - HF\mathbf{s}_{k-1}^i)^T(R+HQH^T)^{-1}(\mathbf{z}_k^j - HF\mathbf{s}_{k-1}^i)] \\ C_k^{i,j} &= (I-KH)Q \\ \mathbf{m}_k^{i,j} &= F\mathbf{s}_{k-1}^i + K(\mathbf{z}_k^j - HF\mathbf{s}_{k-1}^i) \end{aligned} \tag{5.14}$$

and

$$K = QH^T(R+HQH^T)^{-1} \tag{5.15}$$

(5.12) to (5.15) show that given that $p(\mathbf{x}_{k-1}|Z^{k-1})$ is represented by a set of particles with uniform weights $\{\mathbf{s}_{k-1}^i, \frac{1}{N}\}_{i=1}^N$, the *a posteriori* density at time k, $p(\mathbf{x}_k|Z^k)$, is a Gaussian mixture with $N \times (m_k + 1)$ components, and the weight, covariance, and mean of each component are analytically available. It is straightforward to simulate a Gaussian mixture distribution. We draw N uniformly weighted samples according to $p(\mathbf{x}_k|Z^k)$. An update cycle of the proposed algorithm is thus complete.

Two advantages are achieved with the proposed algorithm. Firstly, Monte Carlo simulation allows us, without losing Bayesian optimality, to limit the computational complexity by choosing the number of particles. Secondly, in the standard Sample-Importance-Resample (SIR) particle filter [59, 27], samples are drawn from the *a priori* distribution. When the likelihood is much more peaked than the *a priori* density, this leads to a very low sampling efficiency, since most particles will have negligible weights. Various techniques have been suggested to solve this problem, by "herding" the particles to the right part of the state space [52, 59]. In our case, however, those techniques are not needed. Since we can draw samples directly from the *a posteriori* density, our algorithm is optimal in terms of sampling efficiency [59].

5.4 Using a Second Dynamic Model

When the ball is hit by a player or bounces on the ground, it changes its motion drastically. This may cause a loss of track. To handle the abrupt motion change, a second dynamic model with the following form is also used:

$$F' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.16}$$

When the second dynamic model F' is used, a higher process noise level Q' is assumed. Note that to use the second dynamic model, the only modification needed is to use F' and Q' instead of F and Q in (5.12) to (5.15).

The probability that a particle is propagated with the second dynamic model is determined by the "history" of the particle. At time k, for each particle at time $k-1$, $\mathbf{s}_{k-1}^i$, we search through its history backwards. Assume the first particle in its history that is propagated with the second model is encountered at time k'. If $k - k' > k_{th}$, $\mathbf{s}_{k-1}^i$ is propagated using the second model with a probability of α, and using the first model with a probability of $1 - \alpha$. If $k - k' \leq k_{th}$, the particle is always propagated using the first model.

An example is shown in Fig. 5.4 and Fig. 5.4 of how tracking ambiguity and sudden motion change can be handled by the proposed particle filter. These two figures together show the operation of the particle filter in 48 successive frames. During this period of the time, the near player throws a ball up in the air and serves. The images in Fig. 5.4 and Fig. 5.4 are the relevant areas cropped from the frames.

From frame 67 to 74 (5^{th} and 6^{th} rows of Fig. 5.4), the ball is confused with the net line, and is not detected as a candidate. As a result, the conditional mean of $p(\mathbf{x}_k|Z^k)$ shifts towards the nearby candidates that have originated from clutter. However, due to the constant term β in (5.13), not all particles are "attracted" to the clutter-originated candidates. This allows the recovery of the tracker. As soon as the ball is detected as a candidate again in frame 75 (1^{st} row and 1^{st} column of Fig. 5.4), the tracker "locks on" the ball again.

We can also see in the two figures that, due to the second dynamic model and the higher process noise level associated with it, there are always some "scattered" particles. These particles are critically important when the ball changes its motion abruptly. For example, in frame 87 (4^{th} row and 1^{st} column of Fig. 5.4), the ball is hit by the near player, resulting in an abrupt motion change. However, with the scattered particles, the tracker can still lock onto the ball-originated candidate (and several clutter-originated candidates). All candidates are tracked until the ambiguity is resolved (5^{th} and 6^{th} rows of Fig. 5.4).

One update cycle of the proposed particle filter is summarised in Table 5.4.

5.5 Smoothing

In ball tracking for tennis video annotation, tracking accuracy is important. It has a great impact on the following event detection procedure, and will in turn affect the performance of high level annotation. Estimates computed directly from $p(\mathbf{x}_k|Z^k)$ are usually not accurate enough. This can be seen in Fig. 5.5 (a).

Fig. 5.5 (a) plots the conditional mean of $p(\mathbf{x}_k|Z^k)$, *i.e.* the white crosses in Fig. 5.4 and Fig. 5.4. We can see that this gives poor estimates, especially when the ball is not detected as a candidate for several successive frames (see also the 5^{th} and 6^{th} rows of Fig. 5.4). This is partly due to the fact that $p(\mathbf{x}_k|Z^k)$ does not use any "future" information. In our tennis video annotation

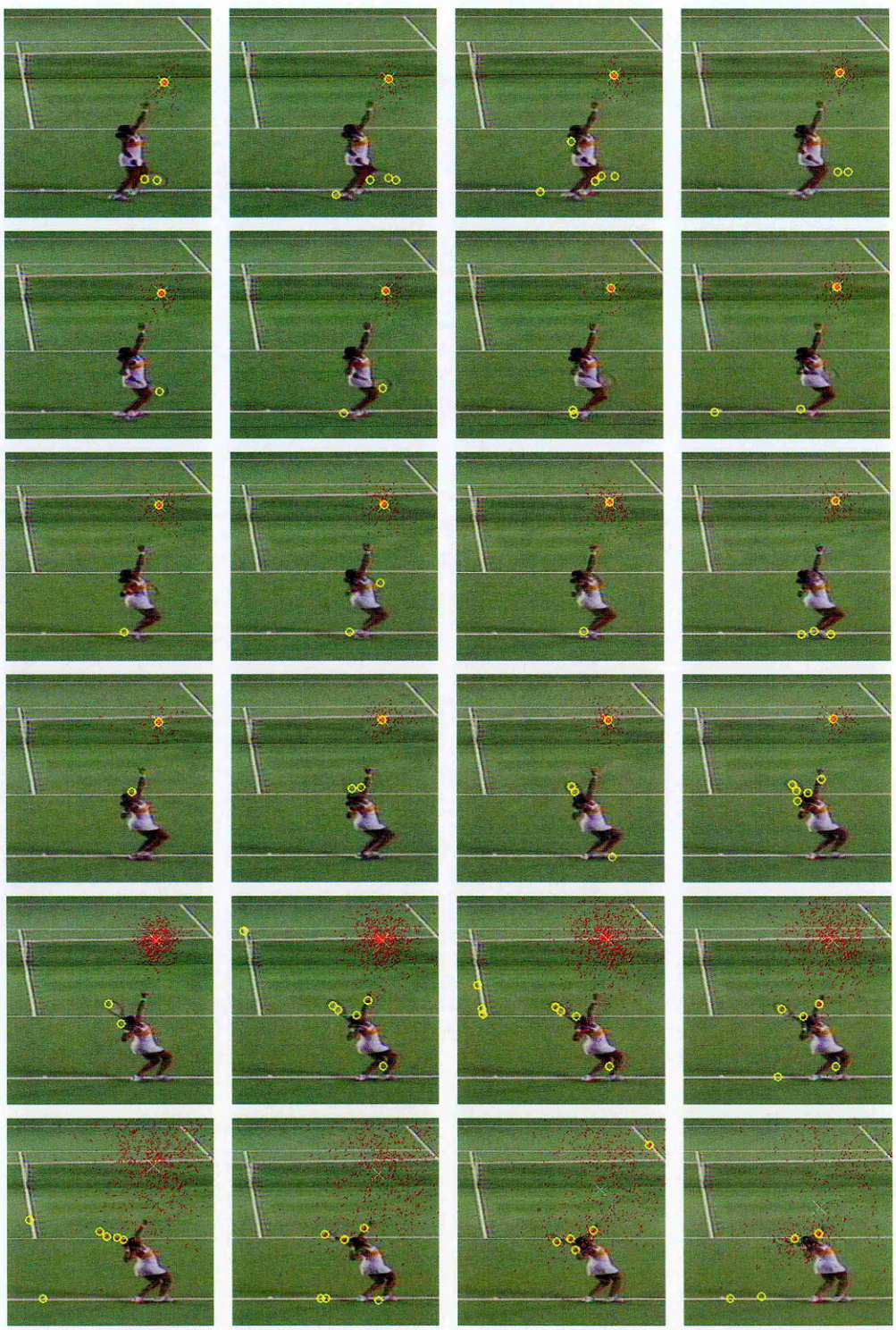


Figure 5.1: Resolving ambiguity and handling sudden motion change with the proposed particle filter - part 1. This figure shows frame 51 to frame 74. In each frame, yellow circles are the ball candidates; red dots are the particles; and white cross is the conditional mean of $p(\mathbf{x}_k|Z^k)$.

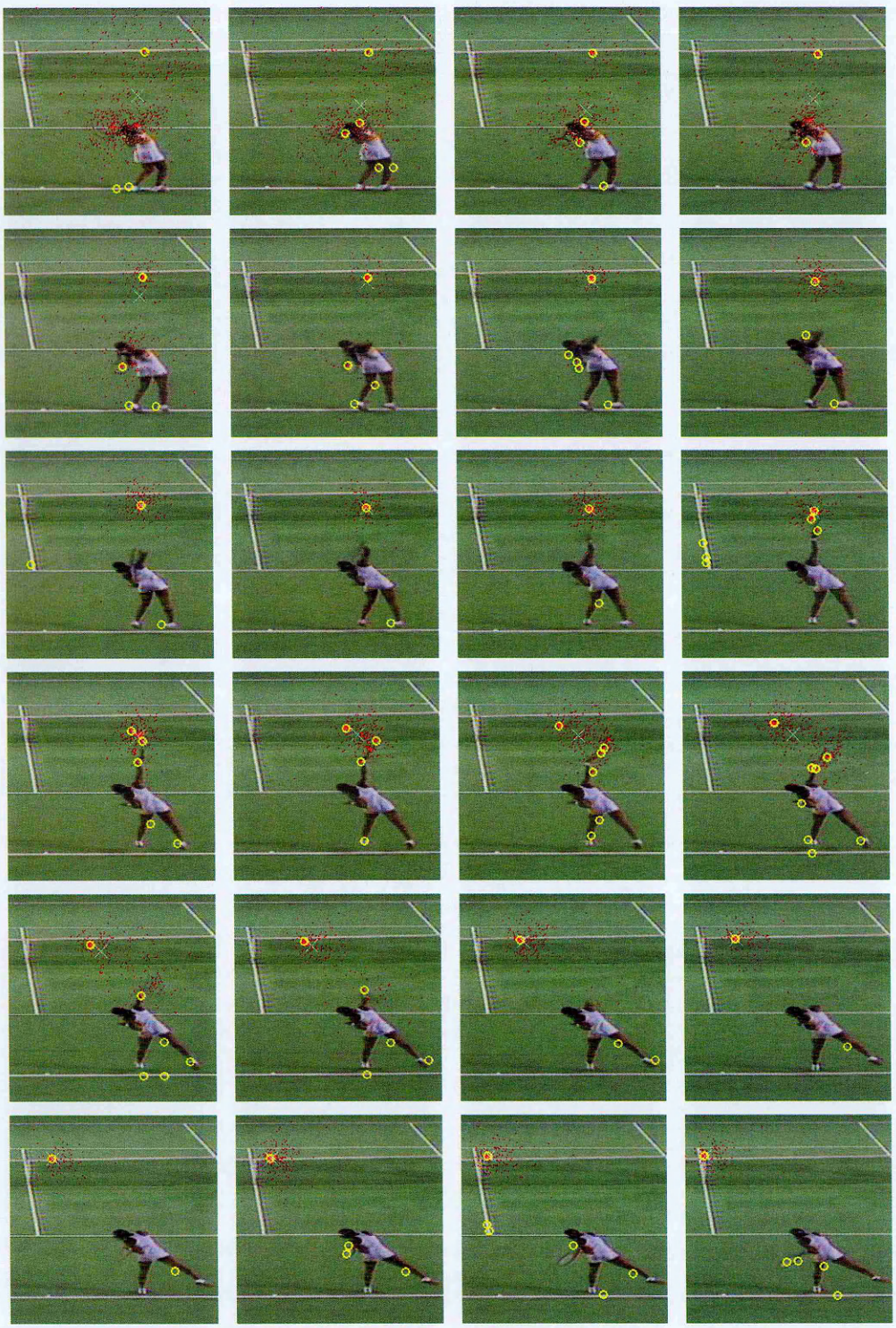


Figure 5.2: Resolving ambiguity and handling sudden motion change with the proposed particle filter - part 2. This figure shows frame 75 to frame 98. In each frame, yellow circles are the ball candidates; red dots are the particles; and white cross is the conditional mean of $p(\mathbf{x}_k|Z^k)$.

Table 5.1: One update cycle of the proposed particle filter.

Input: a set of particles at time $k-1$: $\{\mathbf{s}_{k-1}^i, \frac{1}{N}\}_{i=1}^N$; measurements at time k: $\{\mathbf{z}_k^j\}_{j=1}^{m_k}$.

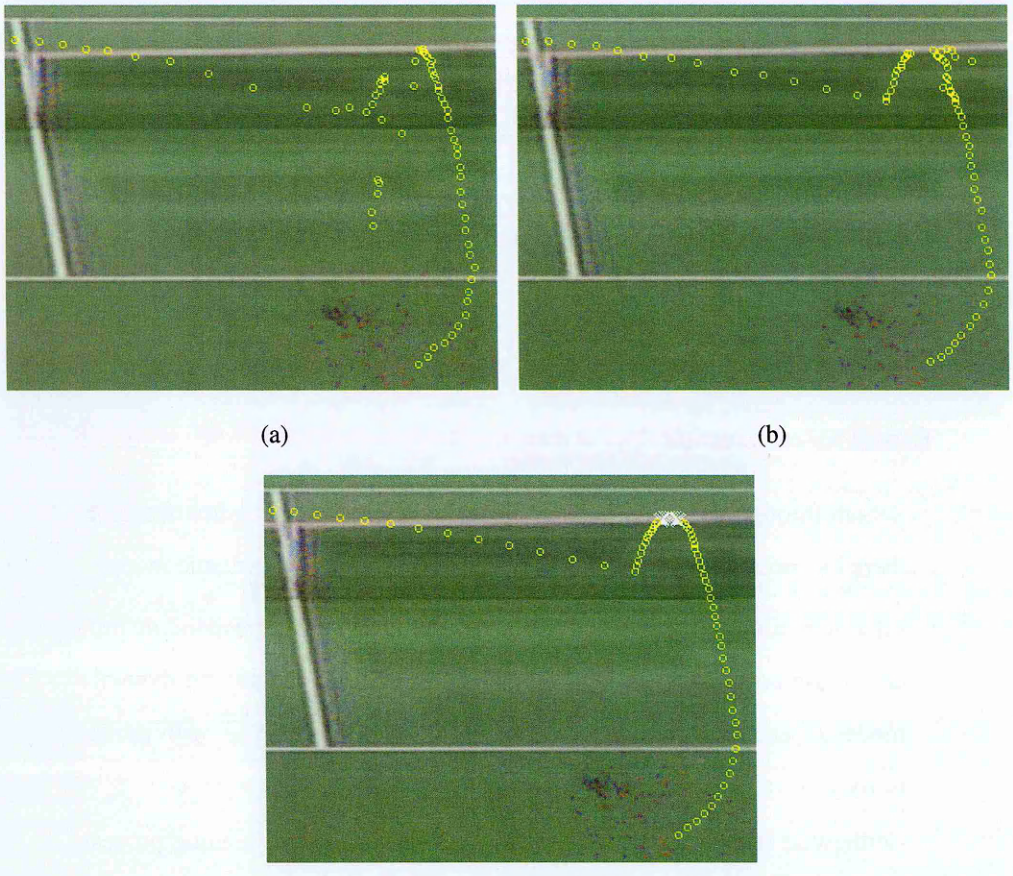
Output: a new set of particles at time k: $\{\mathbf{s}_k^i, \frac{1}{N}\}_{i=1}^N$.

Repeat for each particle $\mathbf{s}_{k-1}^i$ at time $k-1$:

- search through the history of $\mathbf{s}_{k-1}^i$ backwards, and find the first frame k' where its ancestor particle is propagated using the second dynamic model;
  - if $k-k' > k_{th}$, use the first dynamic model F and the corresponding process noise covariance Q with probability α, and use the second dynamic model F' and the corresponding process noise covariance Q' with probability $1-\alpha$;
  - otherwise use the first dynamic model F and the corresponding process noise covariance Q with probability 1;
- compute $w_k^{i,j}$, $C_k^{i,j}$, and $\mathbf{m}_k^{i,j}$ of each component of $p(\mathbf{x}_k|Z^k)$, using (5.13), (5.14) and appropriate dynamic model and process noise level;
- normalise so that $\sum_{i=1}^N \sum_{j=0}^{m_k} w_k^{i,j} = 1$;
- store $w_k^{i,j}$, $C_k^{i,j}$, $\mathbf{m}_k^{i,j}$ and the index i.

Repeat N times:

- select a pair (i,j) with probability $p(i,j) = w_k^{i,j}$;
- sample from the $(i,j)^{th}$ component of the *a posteriori* density. *i.e.* the one with $C_k^{i,j}$ as covariance and $\mathbf{m}_k^{i,j}$ as mean;
- store the sampled particle, along with the index of its parent particle, i, and the dynamic model used, either model 1 or model 2.



(a) (b)

(c)

Figure 5.3: Smoothing and data association. For better visualisation result, we show only the trajectories in the first 100 frames. The trajectories are superimposed on mosaic images. (a): Unsmoothed estimates. Yellow circles are the mean of $p(\mathbf{x}_k|Z^k)$ (unsmoothed density). (b): Smoothed estimates. Yellow circles are the mean of $p(\mathbf{x}_k|Z^K)$ (smoothed density). (c) Final tracking results after data association and interpolation. Yellow circles: ball candidates. White crosses: interpolated positions.

system, ball tracking does not have to be an online process. This means smoothing can be used in a second pass, to estimate $p(\mathbf{x}_k|Z^K)$, *i.e.* the distribution of $\mathbf{x}_k$ in the light of all the measurements in the sequence.

We adopted the fast smoothing algorithm developed in [31]. It is very straightforward. Let $\mathbf{s}_K^i$ be the i^{th} particle at time K, and $\mathbf{s}_k^{i^*}$ be the particle at time k that is in the history of s_K^i. The smoothed density at time k is:

$$p(\mathbf{x}_k|Z^K) = \sum_{i=1}^{N} \delta(\mathbf{x}_k - \mathbf{s}_k^{i^*}) \tag{5.17}$$

The only difference between the unsmoothed density $p(\mathbf{x}_k|Z^k)$ and the smoothed density $p(\mathbf{x}_k|Z^K)$ is that two different sets of supports are used: if a particle at time k is "reinforced" in the future, it has more copies in the smoothed density $p(\mathbf{x}_k|Z^K)$.

The conditional mean of the smoothed density $p(\mathbf{x}_k|Z^K)$ is shown in Fig. 5.5 (b). Comparing Fig. 5.5 (a) and Fig. 5.5 (b), we can see that the latter provides a more accurate estimate of the true state, especially when the true ball is not detected as a candidate (due to confusion with the net line in this case).

5.6 Data Association

Up to this point, the proposed algorithm is state-oriented. That is, it focuses on estimating the **state** of the system, rather than resolving the object-candidate **association** ambiguity. As a result, the data association problem is dealt with implicitly in a "soft" manner. Such a state-oriented approach is not appropriate for our specific application of video annotation, since the reconstructed ball trajectories are usually not accurate enough for the annotation.

This can be seen in Fig. 5.5. The smoothed version of the state density $p(\mathbf{x}_k|Z^K)$ provides a more accurate estimate of the true state than the unsmoothed version $p(\mathbf{x}_k|Z^k)$. However, even the estimates computed from the smoothed density are not accurate during the period when the ball is not detected. This is because for a state-oriented method, the estimates are inevitably "contaminated" by clutter-originated measurements.

There are two sources of estimation inaccuracy in a sequential estimation problem: measurement noise and incorrect object-candidate associations. In our tennis ball tracking problem, the measurement noise is very small, typically under 3 pixels. The second factor is dominant in estimation inaccuracy. In order to get an accurate tracking result, we want to completely eliminate the effect of false candidates.

To completely eliminate the effect of false candidates, we perform an additional data association step after the smoothing. This process explicitly determines whether a candidate is ball-originated or clutter-originated, based on the smoothed density $p(\mathbf{x}_k|Z^K)$.

Using the smoothed density (5.17) and the measurement model (5.2), we have:

$$p(\mathbf{z}_k|Z^K) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)|R|^{\frac{1}{2}}}\exp[-\frac{1}{2}(\mathbf{z}_k - H\mathbf{s}_k^{i*})^T R^{-1}(\mathbf{z}_k - H\mathbf{s}_k^{i*})] \tag{5.18}$$

Substituting $\mathbf{z}_k^j, j = 1, \cdots, m_k$ into (5.18), we get the likelihood of each candidate in the light of all the measurements in the sequence. If the likelihood of at least one candidate is greater than a threshold l_{th}, the candidate with highest likelihood is determined to be ball-originated. If none of the likelihoods is greater than l_{th}, we assume the ball is not detected as a candidate, and its position is interpolated. This process identifies the origin of each candidate, so the effect of false candidates is completely eliminated.

Fig. 5.5 (c) plots the ball trajectory after data association (and interpolation). It should be noted that each yellow circle in Fig. 5.5 (c) is a candidate position, while in Fig. 5.5 (a) and Fig. 5.5 (b) each yellow circle is an estimated position. We can see from Fig. 5.5 (c) that, once the data association ambiguity is resolved, the candidates positions and the interpolated positions are accurate and smooth. Although a Kalman filter or Kalman smoother can be applied to further filter out the process noise and the measurement noise, it is not necessary. In our tennis video annotation system, the trajectory in Fig. 5.5 (c) is used directly for the following modules, such as event detection, without further filtering.

A reconstructed trajectory is a sequence of points in the row-column-time space. In tennis ball tracking, the points at which the ball changes its motion abruptly correspond to key events such as hit or bounce. These events provide important information for high level annotation. We detect these key events by looking for motion discontinuity points in the reconstructed trajectory. The underlying theory is the generalised edge-preserving signal smoothing. Since it is not directly related to the data association problem, the detailed description of the theory is omitted from this thesis, but can be found in [41].

Fig. 5.6 plots the final output of the ball tracking module: a reconstructed trajectory with detected key events. In principle, there may be true events that are not detected (false negatives), and there may also be false events detected (false positives) in the detected events. It is left to the high level reasoning module, where tennis rules are incorporated, to recover form such errors.

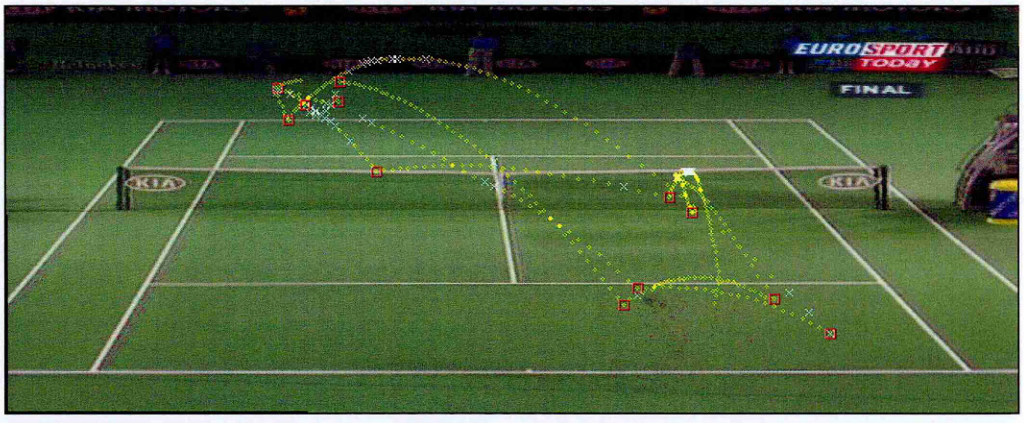


Figure 5.4: Final tracking results (after data association, interpolation and event detection) on the complete sequence superimposed on a mosaic image. Yellow circles: detected ball candidate positions. White crosses: interpolated ball positions. Red squares: detected key events. The tracker goes wrong near the far player, but recovers quickly. This has introduced three false events. In total there are three false positives and two false negatives in event detection. The SVM boundary is -2, which gives a detection rate of $r_d = 0.924$, and an average number of false candidate $\bar{N} = 4.2$ per frame.

5.7 Experiments

Experimental Data and Experiment Setup We used the same 70 sequences as in Chapter 4 for our experiments. Each sequence contains one play starting from a serve. In total the 70 sequences are approximately 8 minutes long, and contain 25,158 frames. First we split the 70 sequences into a training set with 10 sequences containing 3,062 frames, and a test set with 60 sequences containing 22,096 frames. In Chapter 4, we described a ball candidate detection algorithm. It is designed to provide input data for data association. We have also seen in Section 4.4 that setting the SVM boundary -1 seems to provide a good trade-off between the ball detection rate r_d and the average number of false candidate in each frame $\bar{N}$. However, in this chapter, we are interested in the performance of the proposed data association algorithm (the tracker). In order to test the robustness of the tracker in various clutter levels and detection rates, we used six different decision boundaries of the SVM classifier.

The six SVM decision boundaries give six pairs of clutter level and detection rate. For each of the six configurations, we tuned the parameters of the tracker using the training set. During the training, the quantity to minimise was the number of loss-of-track (LOT) frames. To calculate the number of LOT frames, ground truth of the tennis ball positions in all frames was manually

Table 5.2: Proportion of LOT frames.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
prop. of LOT Frames	**2.21%**	**2.26%**	**3.18%**	**12.48%**	**14.11%**	**34.01%**

marked. The tracking results were then compared against the ground truth. We defined the tracking error as the Euclidean distance between the ground truth and the tracked (detected or interpolated) ball position. An LOT frame was defined as a frame where the tracking error was greater than 6 pixels.

Most parameters obtained in the training were constant across the six configurations. The parameters that did vary were the constant term β in (5.8) and the process noise covariance for the second dynamic model Q'. The particle number was fixed to 1000 across all six configurations.

For each configuration, once the optimal set of parameter values was obtained, the tracker was then applied to the 60 sequences in the test set. In order for the tracker to work fully automatically, we have implemented a simple algorithm, which looked for smoothly moving objects, to initialise the tracker. However, in order to measure the performance of the proposed data association algorithm without the effect of track initiation, in our experiments, the tracker was manually initialised.

Distribution of Tracking Errors and Proportion of LOT Frames Fig. 5.7 shows the distribution of tracking errors up to 50 pixels on the test set under various configurations of clutter level and detection rate. Recall that an LOT frame is a frame where the tracking error is greater than 6 pixels, *i.e.* when the tracking error falls outside the first 6 bins in Fig. 5.7. Table 5.7 shows the ratio between the total number of LOT frames and the total number of in-play frames in the 60 sequences.

We can see from Fig. 5.7 and Table 5.7 that the best performance is achieved when the detection rate is high, despite the high clutter level. As the SVM boundary increases, the tracker's performance drops significantly. This can be explained as follows.

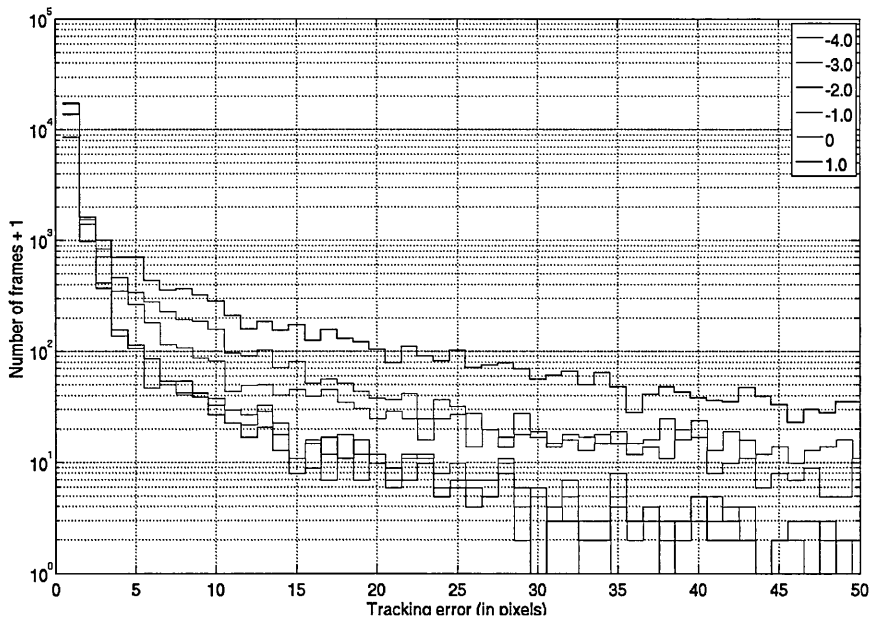
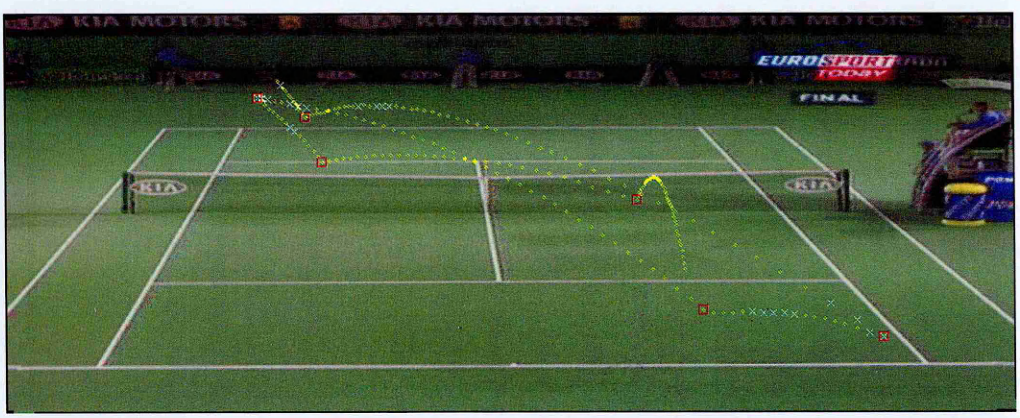


Figure 5.5: Distribution of tracking errors up to 50 pixels. The bin size is 1 pixels. Y-axis is the "number of frames + 1" in logarithmic scale.

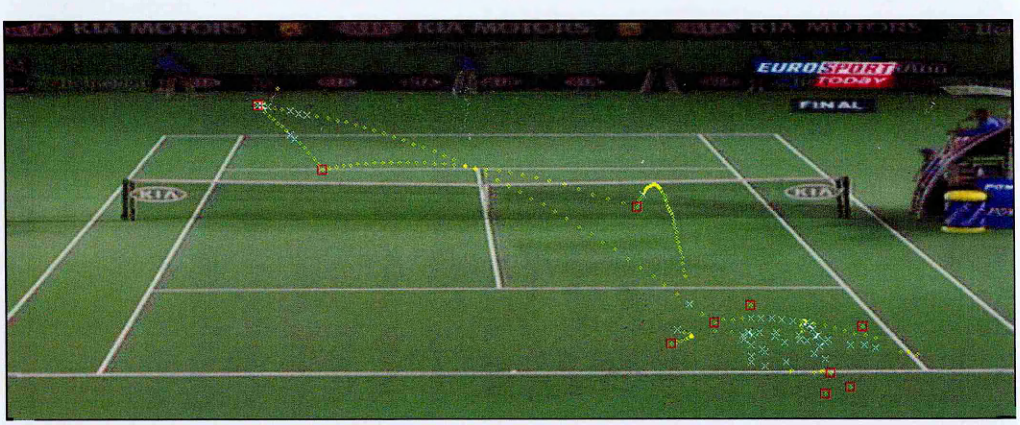
The ball travels at very high velocity after being hit by a player. Due to this high velocity, it appears as an elongated ellipse (see Fig. 1.2 (a)), and departs from its "normal" appearance. As the SVM boundary increases, these elongated ball-originated blobs tend to be wrongly classified as non-candidates. On the other hand, the ball changes its motion drastically after being hit. As a combined result of blob classification error and sudden motion change, the next detected ball-originated candidate can be very far from its predicted position. Although a second dynamic model is used to deal with sudden changes of tennis ball motion, if the next detected ball-originated candidate is too far from its predicted position, the track can still be lost. Once the track is lost, the tracker is usually "trapped" by false candidates that have originated from the player, and can not recover until the ball is close to the player again.

In Fig. 5.7, the peaks towards the right end of the x-axis are caused by loss of track. Fig. 5.7 (b) gives a visual example of what happens when track is lost.

Quality of Event Detection We also measure the performance of the proposed algorithm in terms of quality of event detection. The quality of event detection is obviously affected by the event detection algorithm being used, and thus not a direct quality measure of the ball tracking



(a)



(b)

Figure 5.6: The effect of detection rate on the performance of the tracker. Low detection rate can lead to loss of track, and degrades the performance of the tracker severely. (a) Tracking results when SVM boundary is set to -3. (b) SVM boundary is 0.

Table 5.3: Quality of event detection.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
ground truth	485					
detected	504	497	515	592	812	969
matched	380	386	381	313	292	178
recall	0.784	0.796	0.786	0.645	0.602	0.367
precision	0.754	0.777	0.740	0.529	0.360	0.184
F-measure	**0.769**	**0.786**	**0.762**	**0.581**	**0.450**	**0.245**

Table 5.4: Processing speed.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
frames per sec	**26.3**	**28.0**	**32.1**	**33.8**	**35.5**	**36.3**

Table 5.5: Parameters used in the experiments.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
α	0.5					
standard deviation in Q (pixels)	1.5					
standard deviation in R (pixels)	0.5					
standard deviation in Q' (pixels)	4.0	4.0	4.0	6.0	6.0	8.0
β	0.020	0.020	0.015	0.015	0.010	0.010

algorithm. However, event detection plays a crucial role in high level annotation. We present the quality of event detection here to give an idea of the performance of the proposed algorithm in the context of the tennis annotation system.

Recall that a reconstructed trajectory is a sequence of points in the row-column-time 3D space. For each sequence, points that correspond to a hit or a bounce are manually marked up as ground truth events. We then compare the events detected by the event detection algorithm against the ones in the ground truth. A detected event in a tracked trajectory is regarded as "matched" (true positive) if it is within 3 frames and 5 pixels of a ground truth event. This allows us to calculate the precision and recall of event detection, where the recall is defined as the ratio of the number of matched events to the number of events in the ground truth; and the precision is defined as the ratio of the number of matched events to the number of detected events. We also calculate the F-measure of event detection, which is defined as the harmonic mean of the precision and the recall:

$$F \triangleq \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.19}$$

where F ranges from 0 to 1, and a larger value indicates a better performance.

The precision, recall and F-measure of event detection are shown in Table 5.7. When r_d is low, loss of track happens more frequently. When the track is lost, the ball positions in the reconstructed trajectory are random. Since the event detection algorithm looks for motion discontinuities in a trajectory, a large number of false events are detected, resulting in a very low recall (see Fig. 5.7 (b) for an example).

Processing Speed The processing speed of the proposed data association algorithm is shown in Table 5.7. In the experiments, the number of particles was fixed across the six configurations. However, as we can see in Table 5.7, the processing speed still increases as the clutter level drops. This is because as the number of candidates drops, the number of components in the *a posteriori* density (5.12) also drops, which makes the sampling process faster. Time was measured on a single thread of an Intel Xeon 3.0G computer running Linux, and overhead of disk I/O was included.

5.8 Conclusions

In this chapter, we presented a data association algorithm based on the particle filter. In most particle filtering algorithms, samples are drawn from *a priori* density and evaluated using likelihood density. This can lead to a low sampling efficiency. In our proposed algorithm, samples are drawn directly from the *a posteriori* density. As a result, an improved sampling efficiency is achieved. We also use a second dynamic model to deal with abrupt motion change of the ball. After the filtering pass, smoothing is applied to improve the accuracy of the estimation. Finally, data association is applied, to completely eliminate the effect of false candidates.

Chapter 6

Data Association with the Viterbi Algorithm

6.1 Abstract

The particle filter based algorithm presented in the previous chapter solves the data association problem in an "indirect" way, in that state estimation and object-candidate association are dealt with sequentially. Unlike this "association after estimation" approach, in this chapter, we describe an algorithm that solves the data association problem directly using a modified version of the Viterbi algorithm. The main innovations of this work are: first, using the Viterbi algorithm for searching the sequence of measurements with the maximum *a posteriori* probability; second, a modification to the Viterbi algorithm to allow successive misdetections of the object.

6.2 Problem Formulation

The ball tracking problem is formulated as a sequential estimation problem in the same way as in the previous chapter. A linear, Gaussian and time invariant system is assumed:

$$\mathbf{x}_k = F\mathbf{x}_{k-1} + \mathbf{v} \tag{6.1}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{w} \tag{6.2}$$

where $\mathbf{x}_k$ is the system state, $\mathbf{z}_k$ is the measurement, and $\mathbf{v}$, $\mathbf{w}$ are process noise and measurement noise, respectively. The state vector is the same as defined in (5.3):

$$\mathbf{x}_k = (x_k, y_k, \dot{x}_k, \dot{y}_k, \ddot{x}_k, \ddot{y}_k)^T \tag{6.3}$$

where (x_k, y_k), $(\dot{x}_k, \dot{y}_k)$ and $(\ddot{x}_k, \ddot{y}_k)$ are the position, velocity and acceleration of the ball in the image plane, respectively.

As in the previous chapter, a constant acceleration model is assumed. The state transition matrix F is:

$$F = \begin{pmatrix} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.4}$$

The measurement matrix H is:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.5}$$

In the previous chapter, we proposed a data association algorithm based on particle filtering. It first estimates the system states, then determines the object-candidate association using the estimated states. In this chapter, we propose an algorithm that solves the data association problem directly.

Recall that in the Optimal Bayesian Filter (OBF), the optimal estimate in the Bayesian sense is given by (3.24), but at a price of exponentially growing complexity. There are several ways of controlling the growth of the complexity. The method presented in the previous chapter uses Monte-Carlo simulation, and limits the complexity by controlling the number of particles. On the other hand, non-Monte-Carlo based approaches can be categorised into two groups: combining and pruning. The Mixture Reduction (MR) algorithm and the Probabilistic Data Association (PDA) are two examples of the former group. The idea of combining-based approaches is to limit the ever-growing number of components in the Gaussian Mixture density by combining several components into one. The method we propose in the chapter belongs to the second

group of approaches. Instead of combining components, it tries to prune components that have small probability of being object-originated, and looks for the sequence of object-candidate associations with maximum *a posteriori* probability.

According to (3.23), the *a posteriori* probability of each sequence of candidates being true can be computed recursively. For convenience, we re-write (3.23) as (6.6) below:

$$\begin{aligned}
\beta^{k,l} &\triangleq P\{\Theta^{k,l}|Z^k\} \\
&= P\{\Theta^{k-1,s}, \theta_{k,l}|Z^{k-1}, Z_k\} \\
&\propto P\{Z_k|\theta_{k,l}, \Theta^{k-1,s}, Z^{k-1}\} \times P\{\theta_{k,l}, \Theta^{k-1,s}|Z^{k-1}\} \\
&= P\{Z_k|\theta_{k,l}, \Theta^{k-1,s}, Z^{k-1}\} \times P\{\theta_{k,l}|\Theta^{k-1,s}, Z^{k-1}\} \times P\{\Theta^{k-1,s}|Z^{k-1}\} \\
&= P\{Z_k|\theta_{k,l}, \Theta^{k-1,s}, Z^{k-1}\} \times P\{\theta_{k,l}|\Theta^{k-1,s}, Z^{k-1}\} \times \beta^{k-1,s} \qquad (6.6)
\end{aligned}$$

where all the variables are as defined in Section 3.1: $\Theta^{k,l}$ is the event that $Z^{k,l}$ is the correct sequence of measurements up to time k; $\Theta^{k-1,s}$ is the event that $Z^{k-1,s}$ is the correct sequence of measurements up to time $k-1$; $\theta_{k,l}$ is the event that $\mathbf{z}_{k,l}$ is the object-originated measurement at time k; $Z^{k,l} \triangleq \{Z^{k-1,s}, \mathbf{z}_{k,l}\}$ is the l^{th} sequence of measurements at time k; $Z^{k-1,s}$ is the s^{th} sequence of measurements at time $k-1$; $\mathbf{z}_{k,l}$ is the measurement at time k that is in the l^{th} sequence of measurements; $Z^k \triangleq \{Z^{k-1}, Z_k\}$ is the set of sets of measurements up to time k; Z^{k-1} is the set of sets of measurements up to time $k-1$; and Z_k is the set of measurements at time k. Our goal now is to find the sequence of measurements with maximum *a posteriori* probability, *i.e.* we seek the event

$$\Theta^{k,l^*} \triangleq \arg\max_{\Theta^{k,l}} \beta^{k,l} = \arg\max_{\Theta^{k,l}} P\{\Theta^{k,l}|Z^k\} \qquad (6.7)$$

This is achieved efficiently with a modified version of the Viterbi algorithm.

6.3 The Viterbi Algorithm

Consider a directed graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ is its set of nodes, and $\mathcal{E}$ is its set of directed edges. $\mathcal{G}$ is said to be a trellis if the set of nodes $\mathcal{N}$ can be partitioned into ordered subsets (stages) $\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_K$, such that edges exist only from nodes in stage $\mathcal{N}_{i-1}$ to nodes in stage $\mathcal{N}_i$, where $i = 2, \cdots, K$. Fig. 6.4 gives an example of a trellis, where nodes in the same stage are aligned vertically.

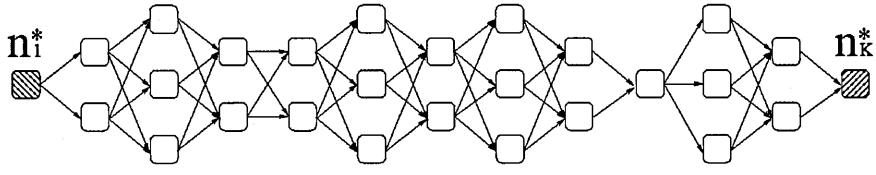


Figure 6.1: An example of a trellis. Nodes in the same stage are aligned vertically. The Viterbi algorithm can be used to find the shortest path between a source node and a destination node (striped nodes in the figure) in an edge-weighted trellis.

Table 6.1: The Viterbi algorithm.

Initialisation at $k = 1$:

- Let $W_1^* = 0$, where W represents the total weight of the path with the smallest total weight among all the paths that terminate at a node;

Repeat for $k = 2, \cdots, K-1$:

- For each node n_k^j in stage k, let $W_k^j = \min(W_{k-1}^i + w_{k-1,k}^{i,j})$, and $\phi_k^j = \arg\min_i(W_{k-1}^i + w_{k-1,k}^{i,j})$, where $i = 1, \cdots, m_{k-1}$;

Termination at $k = K$:

- Let $W_K^* = \min(W_{K-1}^i + w_{K-1,K}^{i,*})$, and $\phi_K^* = \arg\min_i(W_{K-1}^i + w_{K-1,K}^{i,*})$; also let $l_K = *$;

Back tracing for $k = K, \cdots, 2$:

- The label of the node in the shortest path at time $k-1$ is $l_{k-1} = \phi_k^{l_k}$.

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of states in the context of Hidden Markov Model (HMM) [23]. Very briefly speaking, such a problem can be transformed to a shortest path problem in a trellis with its edges appropriately weighed. The Viterbi algorithm can then be used to find the desired shortest path efficiently.

The key idea of the Viterbi algorithm is, for each node at each stage of the trellis, to keep only the "history" and total weight of the shortest path that goes into it. This forward process is repeated until the destination node is reached. The shortest path can then be identified by back tracing. The Viterbi algorithm is summarised in Table 6.3, where the j^{th} node in the k^{th} stage of the trellis is denoted by n_k^j, and the weight of the edge from n_{k-1}^i to n_k^j is denoted by $w_{k-1,k}^{i,j}$.

6.4 Defining the Edge Weight

Having obtained ball candidates in all K frames in a sequence, we construct a trellis in such a way that each frame in the sequence corresponds to a stage in the trellis, and each measurement $\mathbf{z}_k^i$ corresponds to a node n_k^i. An extra "null" node n_k^0 is also introduced at the k^{th} stage of the trellis, which corresponds to the case that the ball is not detected in frame k. Assume that nodes n_1^* and n_K^* are known to correspond to the true object. Now we define the edge weight between two nodes $n_{k-1}^{i_{k-1,s}}$ and $n_k^{i_{k,l}}$ as:

$$w'^{i_{k-1,s},i_{k,l}}_{k-1,k} = -\ln \{P\{Z_k|\theta_{k,l}, \Theta^{k-1,s}, Z^{k-1}\} \times P\{\theta_{k,l}|\Theta^{k-1,s}, Z^{k-1}\}\} \qquad (6.8)$$

where $i_{k,l}$ is the label of the candidate in frame k that is in the l^{th} sequence of measurements at time k; and $i_{k-1,s}$ is the label of the candidate in frame $k-1$ that is in the s^{th} sequence of measurements at time $k-1$. From (6.6), (6.7) and (6.8), the shortest path given by the Viterbi algorithm will correspond to the sequence of measurements with maximum *a posteriori* probability. It should be noted that according to the definition of edge weight in (6.8), each edge can have multiple weights, depending on the "history" of the path being considered. In other words, our problem of searching the maximum *a posteriori* probability sequence of measurements is **not** Markovian. However, as we will see shortly, the dynamic programming principle of the Viterbi algorithm can still be applied to approximate the shortest path.

Now we derive the edge weight in (6.8). By assuming false candidates are uniformly distributed

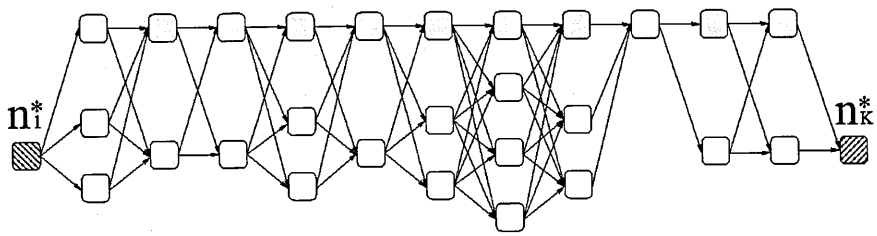


Figure 6.2: The data association problem mapped onto a trellis. Each non-shaded node is a ball candidate, and each shaded node is a "null" node, representing the case where the ball is not detected as candidate in the frame. Nodes in the same stage are aligned vertically. Assume the first and the last ball-originated candidates have been identified (the striped nodes), and the edge weights are defined as in (6.11), the Viterbi algorithm can be used to approximate the shortest path between the two striped nodes, which corresponds to the sequence of measurements with maximum *a posteriori* probability.

in the image plane, the likelihood term in (6.8) is [2]:

$$P\{Z_k|\theta_{k,l},\Theta^{k-1,s},Z^{k-1}\} = \begin{cases} V^{-(m_k-1)}\mathcal{N}[\mathbf{z}_{k,l};\hat{\mathbf{z}}^s(k|k-1),\mathbf{S}^s(k)] & i_{k,l}\neq 0 \\ V^{-m_k} & i_{k,l}=0 \end{cases} \tag{6.9}$$

and the *a priori* probability term in (6.8) is [2]:

$$P\{\theta_{k,l}|\Theta^{k-1,s},Z^{k-1}\} = P\{\theta_{k,l}|m_k\} = \begin{cases} \frac{1}{m_k}r_d & i_{k,l}\neq 0 \\ 1-r_d & i_{k,l}=0 \end{cases} \tag{6.10}$$

where m_k is the number of candidates in frame k, V is the "volume" of the whole image plane; r_d is the object detection probability as defined in (4.11); $\hat{\mathbf{z}}^s(k|k-1)$ is the predicted measurement given by the Kalman filter that is associated with the s^{th} sequence of measurements at time $k-1$; $\mathbf{S}^s(k)$ is the corresponding innovation covariance matrix. Taking the product of (6.9) and (6.10), and after some rearrangements and normalisation, the final definition of edge weight is obtained as:

$$w_{k-1,k}^{i_{k-1,s},i_{k,l}} = \begin{cases} -\ln\{\mathcal{N}[\mathbf{z}_{k,l};\hat{\mathbf{z}}^s(k|k-1),\mathbf{S}^s(k)]\} & i_{k,l}\neq 0 \\ -\ln\{\frac{1-r_d}{r_d}\frac{m_k}{V}\} & i_{k,l}=0 \end{cases} \tag{6.11}$$

6.5 Modifying the Viterbi Algorithm to Allow Successive Misdetections

Having defined the edge weights in the trellis, the Viterbi algorithm can be applied to find the shortest path between n_1^* and n_K^*, *i.e.* the sequence of measurements with the maximum *a poste-*

riori probability of being ture. Recall that we assume the first and last ball-originated candidates have been identified, and the corresponding measurements are $\mathbf{z}_1^*$ and $\mathbf{z}_K^*$, respectively. To start with, the set of "surviving" Kalman filters at time 1, $\mathcal{F}_1$, is empty. We initialise a Kalman filter with $\mathbf{z}_1^*$, and insert it into $\mathcal{F}_1$. The forward process of the Viterbi algorithm is then performed. At each time step k, we calculate the edge distances using each pair of Kalman filter in $\mathcal{F}_{k-1}$ and measurement in Z_k. In the standard Viterbi algorithm, we would like to keep, for each node, only the Kalman filter associated with the measurement history that has the smallest total weight, and insert these "surviving" Kalman filters to $\mathcal{F}_k$. In our tennis ball tracking problem, however, due to occlusion, motion blur, *etc.*, the ball may not be detected as a candidate in several successive frames. In order to cope with successive misdetections, while still being able to control the complexity of the algorithm, the standard Viterbi algorithm is modified as follows.

In the forward recursion process of the Viterbi algorithm, at each time step k, the pruning of non-optimal paths is performed only for non-null nodes, that is, nodes n_k^j where $j = 1, \cdots, m_k$. For the null node n_k^0, a different pruning scheme is used. First, a "life time" parameter N_{life} is set for each surviving Kalman filter in $\mathcal{F}_{k-1}$, which represents the number of "null-node frames" a Kalman filter can tolerate before it is discarded. N_{life} is set in such a way that a Kalman filter associated with a measurement history with a smaller total weight has a greater N_{life}. At time k, we check the set of surviving Kalman filters in $\mathcal{F}_{k-1}$. If a Kalman filter in $\mathcal{F}_{k-1}$ contains more than N_{life} successive null nodes in its recent history, it is discarded; otherwise it is updated with a null measurement, and the updated Kalman filter is inserted to $\mathcal{F}_k$.

This forward recursion is repeated until the destination node n_K^* is reached. After pruning, there is only one surviving Kalman filter in $\mathcal{F}_K$. According to the definition of edge weights, the sequence of measurements associated with the only surviving Kalman filter has the maximum *a posteriori* probability. The complete Viterbi-based data association algorithm is summarised in Table 6.5. Fig. 6.5 illustrates the algorithm on an example sequence.

6.6 Experiments

Experimental Data and Experiment Setup We used the same experimental data as in the previous chapter for evaluating the performance of the Viterbi-based data association algorithm.

Table 6.2: Data association with a modified version of the Viterbi algorithm.

Initialisation at $k = 1$:

- initialise a Kalman filter with $\mathbf{z}_1^*$, and put it into the set of surviving Kalman filters at time 1, $\mathcal{F}_1$; also let $W_1^* = 0$;

Repeat for $k = 2, ...K-1$:

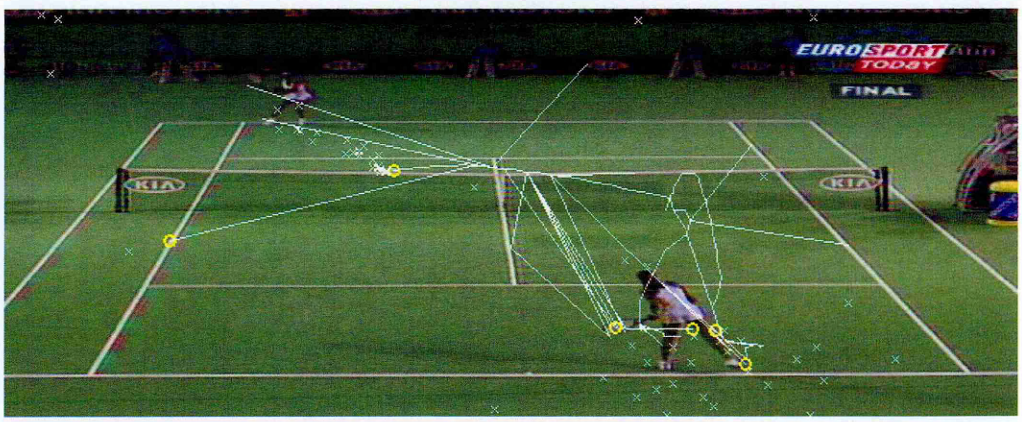
- for each node $n_k^{i_{k,l}}$ in frame k, calculate $w_{k-1,k}^{i_{k-1,s},i_{k,l}}$ using $\mathbf{z}_k^{i_{k,l}}$ and each Kalman filter in $\mathcal{F}_{k-1}$, according to (6.11); note that each Kalman filter is associated with a path in the trellis, and vice versa;

  - for each $n_k^{i_{k,l}}, i_{k,l} \neq 0$, among all the paths that terminate at it, find the one that has the lowest total weight, $W_k^{i_{k,l}} = \min(W_{k-1}^{i_{k-1,s}} + w_{k-1,k}^{i_{k-1,s},i_{k,l}})$; update the corresponding Kalman filter with $\mathbf{z}_k^{i_{k,l}}$, and insert the updated Kalman filter to $\mathcal{F}_k$;

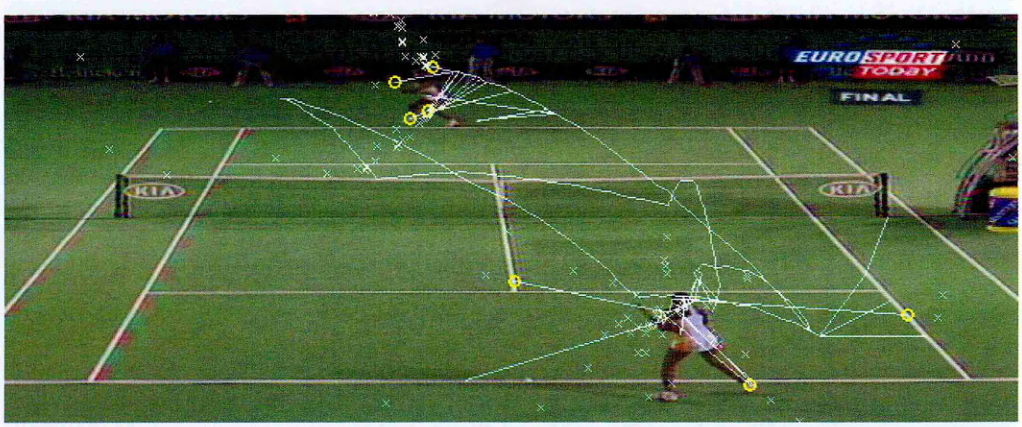
  - for node n_k^0, if a path that terminates at it contains more than N_{life} successive null nodes in its recent history, the path is discarded; otherwise the corresponding Kalman filter is updated with a null measurement, and let $W_k^0 = W_{k-1}^{i_{k-1,s}} + w_{k-1,k}^{i_{k-1,s},0}$; the updated Kalman filter is inserted to $\mathcal{F}_k$;

Termination at $k = K$:

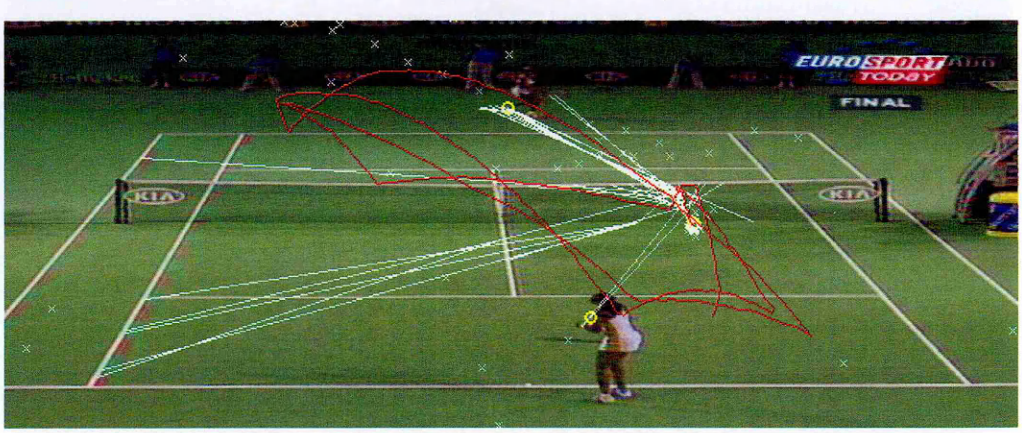
- Among all the paths that terminate at n_K^*, find the one that has the lowest total weight. This path corresponds to the sequence of measurements with maximum *a posteriori* probability.



(a)



(b)



(c)

Figure 6.3: (a) and (b): the Viterbi-based data association algorithm at time step 105 and 224, respectively. Lines: paths associated with the "surviving" Kalman filters. Crosses: predicted ball positions given by the "surviving" Kalman filters. (c) The termination step. The red line shows the sequence of measurements with the maximum *a posteriori* probability.

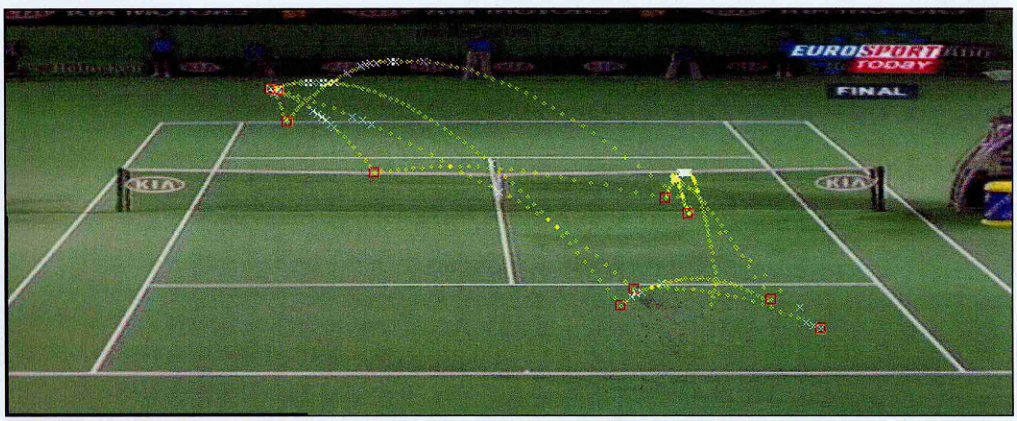


Figure 6.4: Final tracking results (after interpolation and event detection) superimposed on the mosaic image. Yellow circles: detected ball candidate positions. White crosses: interpolated ball positions. Red squares: detected key events.

The data set consists of 70 tennis sequences, with one play in each sequence. As in the previous chapter, the 70 sequences were split into a training set with 10 sequences and a test set with 60 sequences. For each SVM boundary, parameters of the tracker were tuned using the training set. The tracker with the optimal set of parameters was then evaluated using the test set. Also as in the previous chapter, the tracker was manually initialised, in order to eliminate the effect of track initiation and track termination.

Distribution of Tracking Errors and Proportion of LOT Frames Fig. 6.6 and Table 6.6 show the distribution of tracking errors up to 50 pixels and the proportion of LOT frames on the test set under various configurations of clutter level and detection rate, respectively. From Fig. 6.6 and Table 6.6, we can see that as the SVM boundary increases from -4 to -2, the performance of the tracker improves. This is because during this time, the average number of false candidates $\bar{N}$ drops significantly, while the detection rate r_d drops only slightly.

As the SVM boundary continues to increase from -2 to 1, the performance of the tracker drops, as in the particle filter based algorithm. However, the degradation of the performance is not as severe as in the particle filter based algorithm. This is because the two algorithms have different failure modes under low detection rate. As discussed in the previous chapter, in the particle filter based data association algorithm, a low detection rate can lead to loss of track, and thus severely degrades the algorithm's performance. On the other hand, by tuning the "life time"

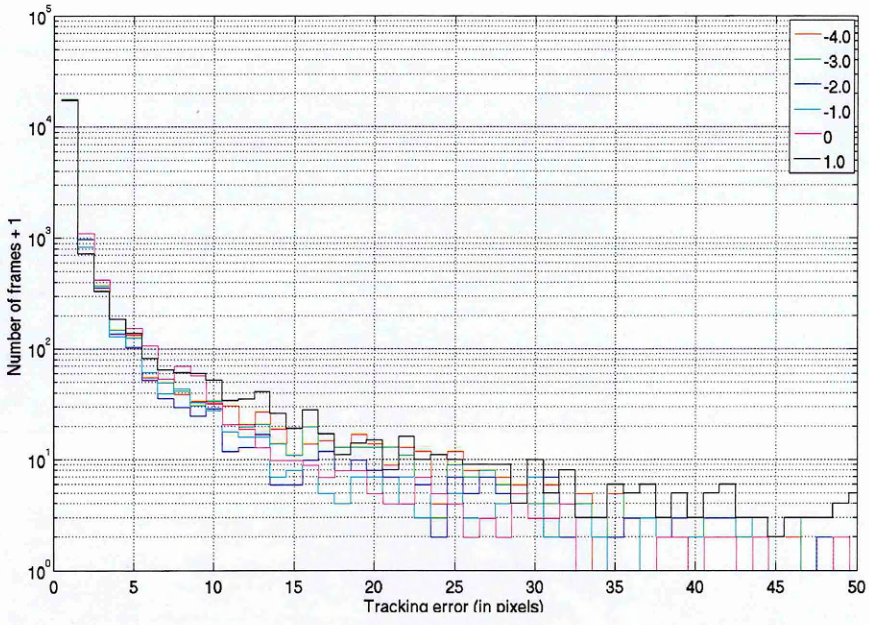


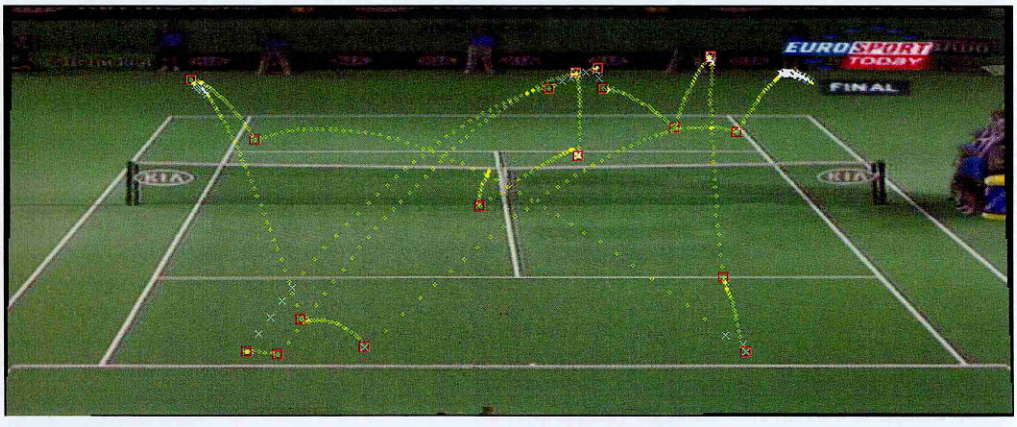
Figure 6.5: Distribution of tracking errors up to 50 pixels. The bin size is 1 pixels. Y-axis is the "number of frames + 1" in logarithmic scale.

Table 6.3: Proportion of LOT frames.

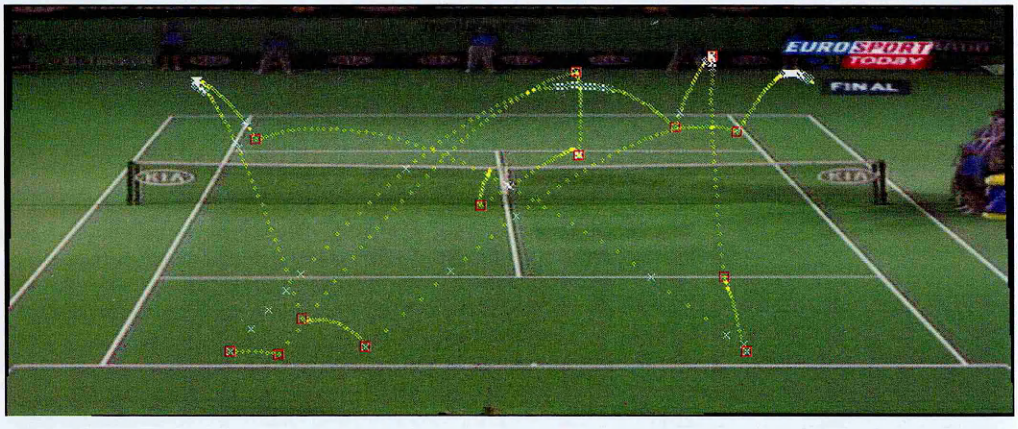
r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
prop. of LOT Frames	**2.27%**	**2.08%**	**1.44%**	**1.51%**	**1.90%**	**4.11%**

parameter N_{life} during the training process, the Viterbi based algorithm can survive successive misdetections. As a result, it is much more robust to low detection rate. In the Viterbi based algorithm, the increase of tracking error as r_d drops is caused by inaccuracy of interpolation instead of loss of track, as shown in Fig. 6.6.

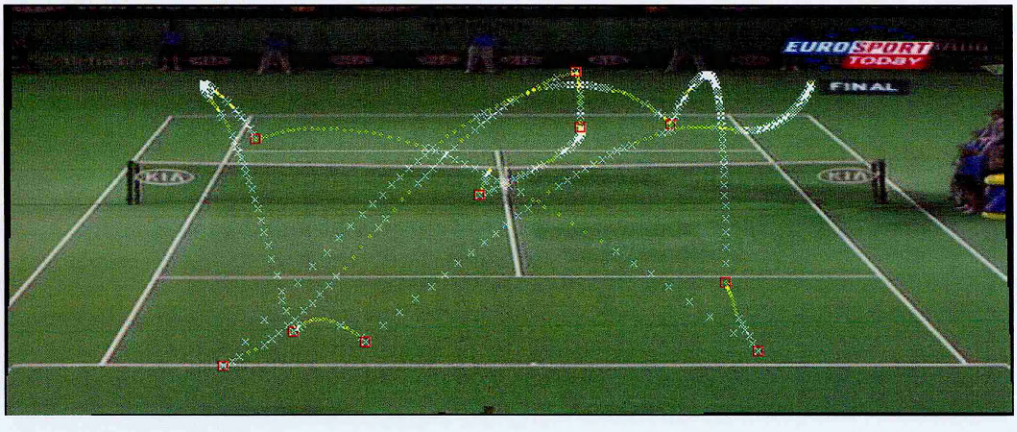
Quality of Event Detection The performance of the algorithm in terms of the quality of event detection is shown in Table 6.6. Similar trend as in LOT analysis is observed: as the SVM boundary increases, the F-measure of event detection first increases then drops.



(a)



(b)



(c)

Figure 6.6: The effect of the SVM boundary on the tracker's performance. From (a) to (c): the SVM boundary is -4, -2, and 1, respectively. As the detection rate and clutter level increase, the tracker's performance first improves then drops.

Table 6.4: Quality of event detection.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
ground truth	485					
detected	487	493	467	464	410	364
matched	382	386	387	389	337	274
recall	0.788	0.796	0.798	0.802	0.695	0.565
precision	0.769	0.783	0.829	0.838	0.822	0.753
F-measure	**0.778**	**0.789**	**0.813**	**0.820**	**0.753**	**0.646**

Table 6.5: Processing speed.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
frames per sec	**42.3**	**46.3**	**50.1**	**51.2**	**52.0**	**53.4**

Table 6.6: Parameters used in the experiments.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
standard deviation in Q (pixels)	1.5					
standard deviation in R (pixels)	0.5					
N_{life} (frames)	$1 \sim 8$	$1 \sim 10$	$1 \sim 12$	$1 \sim 16$	$1 \sim 20$	$1 \sim 25$

Processing Speed The processing speed of the proposed data association algorithm is shown in Table 6.6. Clearly, the processing speed increases as the clutter level drops. This is because the size of the trellis diagram is determined by the number of candidates. More false candidates will increase both the number of nodes in the trellis and the number of surviving Kalman filters in each time step, thus the increased time complexity.

6.7 Conclusions

In this chapter, we proposed a data association algorithm based on the Viterbi algorithm. Unlike the particle filter based algorithm described in the previous chapter, which solves the estimation problem and the data association problem sequentially, the algorithm presented in this chapter directly seeks the sequence of object-measurement associations with maximum *a posteriori* probability. This problem is transformed to a shortest path problem in a trellis diagram with its edge weights appropriately defined, and the desired shortest path is obtained efficiently using a modified version the Viterbi algorithm.

Chapter 7

Layered Data Association with Graph-Theoretic Formulation

7.1 Abstract

In this chapter, we propose a multi-layered data association algorithm with graph-theoretic formulation. Unlike the two algorithms described in the previous chapters, this algorithm is fully automatic, and is capable of tracking multiple objects. At the object candidate level, "tracklets" are "grown" from sets of candidates that have high probabilities of containing only true positives. The innovation at this level is an extension to the RANdom SAmple Consensus (RANSAC) algorithm, which results in significant efficiency improvement over the baseline RANSAC algorithm. At the tracklet level, a directed and weighted graph is constructed, where each node is a tracklet, and the edge weight between two nodes is defined according to the "compatibility" of the two tracklets. The innovation at this level is to formulate the association problem as a shortest path problem in this graph. By exploiting a special topological property of the graph, we have also developed a more efficient all-pairs shortest path (APSP) algorithm than the general-purpose ones. Finally, at the path level, the innovation is to us a path reduction (PR) algorithm to reduce the set of all-pairs shortest paths to an best set of compatible paths (BSCP), which corresponds to the set of ball trajectories.

7.2 Overview of the Strategy

To better appreciate the data association problem at hand, we plot in Fig. 7.2 all the ball candidates in a sequence in a column-row-time 3D space. In Fig. 7.2, blue circles denote clutter-originated candidates, and circles with other colours denote ball-originated candidates. Semantically, the ball-originated candidates belong to three plays. In time order (from bottom to top in the figure), the first play (magenta circles) is a bad serve, where the ball lands outside the service box; the second "play" (cyan circles) is a player bouncing the ball on the ground preparing for the next serve; and the third play (red circles) is a relatively long one with several exchanges. The objective of data association is to identify the number of plays in this sequence, and identify all the ball-originated candidates in each play. In other words, the objective is to recover the colour information in Fig. 7.2, assuming it is lost. As discussed earlier, once this is achieved, the estimation problem, if still desired, is trivial. It should be noted that the second "play" in this sequence is not a real play with serve, ball exchange, *etc.* However, as far as ball tracking is concerned, it forms a perfectly valid ball trajectory. In our tennis video annotation system, it is left to the higher level module to distinguish such a "play" from regular ones, when more information, such as player position and player action, is available.

Fig. 7.2 provides a "global view" of the data association problem. It reminds us that our task of ball trajectory reconstruction is an off-line application, which means we have access to "future" information when resolving data association ambiguity. Naturally, not only should we exploit this future information for deciding the association, but also we should exploit it as "fully" as possible. The particle filter based algorithm in Chapter 5 exploits future information by applying smoothing as a second pass. The Viterbi based algorithm in Chapter 6 utilises future information by delaying the data association decision until the destination node is reached in the forward recursion pass. Both algorithms somehow make use of future information one way or another, but not necessarily in the fullest and most effective ways.

Now let us recall the Robust Data Association (RDA) algorithm reviewed in Chapter 3. The key idea of RDA is to treat data association as a motion model fitting problem. By using the RANSAC paradigm, robustness against clutter is achieved. On the other hand, several deficiencies of RDA have been noticed. Firstly, RDA is a single-object tracking algorithm, and by itself cannot deal with track initiation and track termination. This means RDA is not suitable

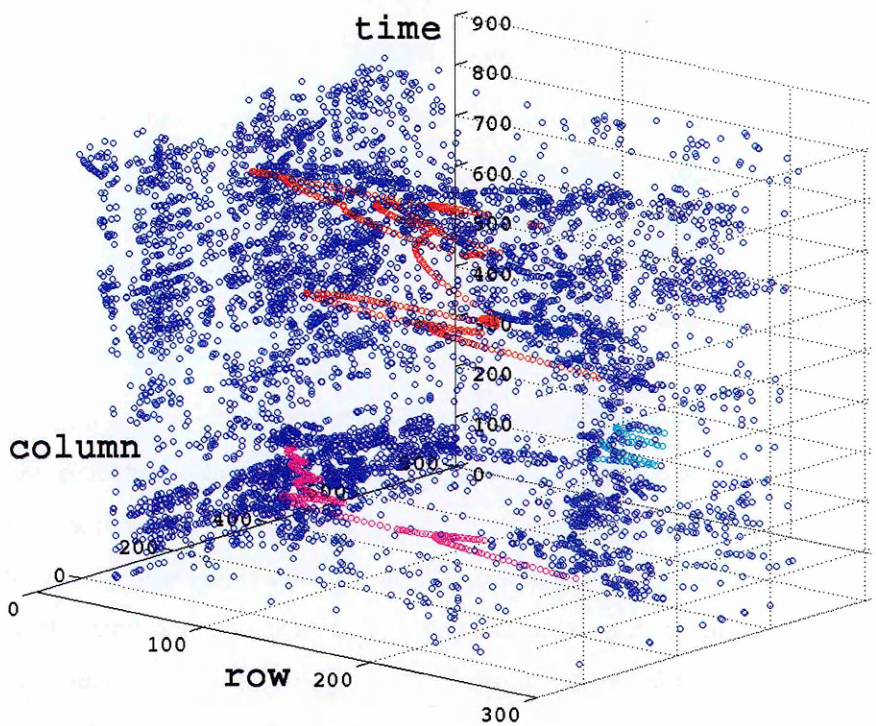


Figure 7.1: All ball candidates in a sequence plotted in a column-row-time 3D space. The SVM boundary is -4. Each circle in the figure is a ball candidate. Blue circles: clutter-originated candidates. Circles in other colours: ball-originated candidates. The magenta, cyan, and red circles correspond to the first, second and third play in the sequence, respectively. The objective of tennis ball tracking is to recover the class label of each candidate: clutter, first play, second play, or third play.

for sequences that have complex track initiation/termination of multiple balls. Secondly, like most RANSAC-based algorithms, in RDA, samples are drawn randomly from all candidates in an interval. As the proportion of true positives drops, the number of trials required to guarantee a certain probability of getting an "uncontaminated" sample set increases fast. This fast growing complexity makes RDA impractical in highly cluttered environments. Lastly, in RDA, estimates are given by the best models in corresponding intervals independently of each other. No motion smoothness constraint is applied. If a clutter-originated motion is wrongly picked up as the best model in an interval, there is no mechanism to recover from such an error.

Inspired by the conclusion that all available information should be exploited as fully as possible for decision making, and by the RDA's model-fitting approach to the data association problem, in this chapter we propose a three-layer data association scheme with graph-theoretic formulation. From bottom to top, the three layers are: candidate level association, tracklet level association, and path level analysis.

I. Assume ball candidates in each frame have been generated. Also assume a sliding window is moved over the sequence. **At the candidate level,** instead of randomly sampling as in RDA, we exhaustively evaluate for each candidate in the middle frame of the sliding window whether a small ellipsoid around it in the column-row-time space contains one candidate from the previous frame and one candidate from the next frame. If it does, we call the 3 candidates inside the ellipsoid a "seed triplet", and fit a dynamic model to it. The fitted model is then optimised recursively using candidates in the sliding window that are consistent with it. This process is repeated until convergence, forming what we call a "tracklet". This "hill-climbing" scheme significantly reduces the algorithm's complexity: as the proportion of true positives drops, the complexity grows approximately linearly.

II. As the sliding window moves, a sequence of tracklets is generated. These tracklets may have originated from different balls or from clutter. Now we need a data association method **at the tracklet level**. We formulate tracklet level association as a shortest path problem. We construct a weighted and directed graph, where each node is a tracklet, and the edge weight between two nodes is defined according to the "compatibility" of the two tracklets. If we were to assume that there was only one ball to track, and that the first and last tracklets that have originated from this ball were already known, the shortest path between these two

tracklets (nodes) in the graph would then correspond to the trajectory of the ball.

III. The above shortest path formulation assumes that there is only one ball to track, and that the first and last tracklets of this ball are known. We relax these assumptions by looking for shortest paths between all pairs of nodes in the graph, instead of the shortest path between a given pair of nodes. By analysing the all-pairs shortest paths **at the path level**, the first and last tracklets of each ball can be identified. Track initiation/termination of this multiple object tracking problem is then automatically dealt with. A fully automated data association algorithm is thus complete.

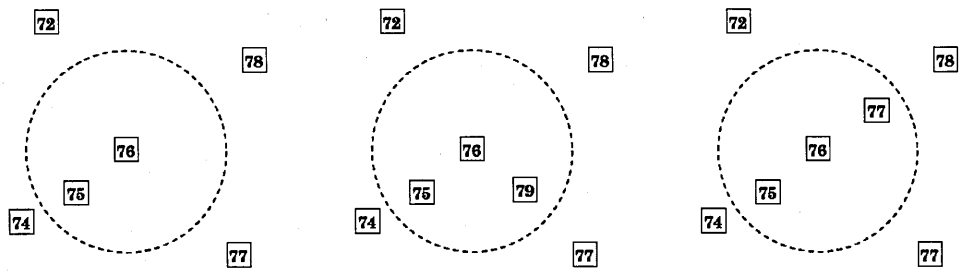
The rest of this chapter is organised as follows: Section 7.3, Section 7.4 and Section 7.5 describe the three layers of the proposed scheme: candidate level association, tracklet level association, and path level analysis, respectively. Section 7.6 discusses processing speed considerations. An efficient all-pairs shortest path algorithm is also presented in this section. Experimental results are presented in Section 7.7 and Section 7.8. Finally, conclusions are given in Section 7.9.

7.3 Candidate Level Association

In this section, we describe the bottom layer of the proposed scheme. We show how to "grow" ball candidates into "tracklets" — small segments of trajectories. This process starts with a "seed triplet", a set of three candidates closely clustered in the row-column-time 3D space. A dynamic model is then fitted to the seed triplet, and optimised recursively to form a tracklet. After this process, the data association problem is transformed from the candidate space to a tracklet space.

7.3.1 Looking For a Seed Triplet

As usual we assume a sequence is composed of K frames, and the frames are numbered from 1 to K. Let us denote the set of candidates in frame k by $\mathcal{C}_k = \{c_k^j\}_{j=1}^{m_k}$, where m_k is the number of candidates in frame k, and c_k^j is the j^{th} candidate in $\mathcal{C}_k$. The corresponding set of measurements, *i.e.* positions of the candidates, are denoted by $Z_k = \{z_k^j\}_{j=1}^{m_k}$ as usual. Assume a sliding window containing $2V+1$ frames is moving upward along the time axis in Fig. 7.2.



(a) (b) (c)

Figure 7.2: Three examples of looking for seed triplet. Squares with numbers: candidates detected in different frames. Dashed circle: the circular area A_i^j around a candidate c_i^j. The radius of this circle is R. (a) and (b) For the candidate in frame 76, no seed triplet is found. Note that in (b), although 2 candidates (besides the one from frame 76) fall into the circular area, there is no candidate from C_{77}. (c) Sufficient candidates are found in the circular area to form a seed triplet: one from C_{75} and one from C_{77}.

At time i, the interval I_i spans frame $i - V$ to frame $i + V$. We project all the candidates inside interval I_i, *i.e.* all the candidates in $\mathcal{C}^{(i)} \triangleq \{C_{i-V}, ..., C_{i+V}\}$ onto the bottom plane in Fig. 7.2, *i.e.* the row-column image plane. In this image plane, centred at each $c_i^j \in C_i$ and with radius R, a circular area A_i^j is considered, where R is the maximum distance the ball can travel between two successive frames. We examine whether at least one candidate from C_{i-1} and at least one candidate from C_{i+1} fall into A_i^j (see Fig. 7.3.1). Assume $c_{i-1}^{j'} \in C_{i-1}$ and $c_{i+1}^{j''} \in C_{i+1}$ are found inside A_i^j, throughout the rest of this paper, we call the 3 candidates $c_{i-1}^{j'}$, c_i^j and $c_{i+1}^{j''}$ a **seed triplet.**

7.3.2 Fitting a Model to the Seed Triplet

Now consider any 3 candidates detected in frame k_1, k_2 and k_3, where $k_1 < k_2 < k_3$. Let the positions of the three candidates be $\mathbf{z}_1$, $\mathbf{z}_2$ and $\mathbf{z}_3$, respectively. A constant acceleration model M can be solved as:

$$\mathbf{v}_1 = \frac{\mathbf{z}_2 - \mathbf{z}_1}{\Delta k_{21}} - \frac{\Delta k_{21} \times \mathbf{a}}{2} \tag{7.1}$$

$$\mathbf{a} = 2 \times \frac{\Delta k_{21} \times (\mathbf{z}_3 - \mathbf{z}_2) - \Delta k_{32} \times (\mathbf{z}_2 - \mathbf{z}_1)}{\Delta k_{21} \times \Delta k_{32} \times (\Delta k_{21} + \Delta k_{32})} \tag{7.2}$$

where $\Delta k_{21} \triangleq k_2 - k_1$, $\Delta k_{32} \triangleq k_3 - k_2$, $\mathbf{a}$ is the constant acceleration, $\mathbf{v}_1$ is the velocity at time k_1, and $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{a}, \mathbf{v}_1 \in \mathbb{R}^2$. An estimate of the ball position in any frame k is then given

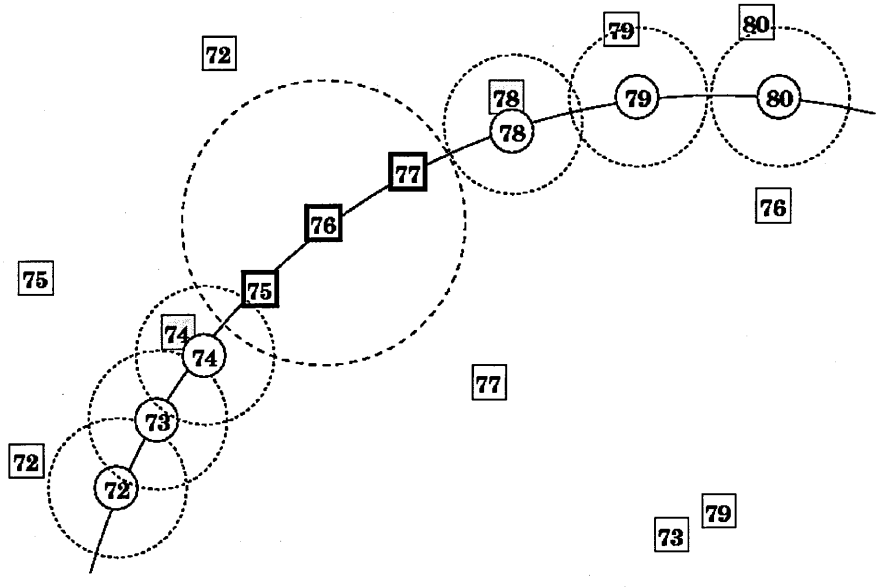


Figure 7.3: Fitting a motion model to the seed triplet in Fig. 7.3.1 (c). Squares with numbers: candidates detected in different frames. Big dashed circle: the circular area A_i^j around the candidate c_i^j. The radius of this circle is R. Bold squares: the triplet used for model fitting. Solid circles with numbers: positions estimated with the fitted model. Small dashed circles: the region a candidate has to fall into to be a support. The radius of these circles is d_{th}. Shaded squares: candidates that are in the support set of the model after the current iteration. Note that the bold squares are also in the support set.

by

$$\hat{\mathbf{z}}_k = \mathbf{z}_1 + \Delta k \times \mathbf{v}_1 + \frac{(\Delta k)^2}{2} \times \mathbf{a} \tag{7.3}$$

where $\Delta k \triangleq k - k_1$.

Such a model can be fitted to any 3 candidates detected in different frames. Now we apply it to the seed triplet found inside A_i^j, as illustrated in Fig. 7.3.2. In this special case, $k_1 = i - 1$, $k_2 = i$, and $k_3 = i + 1$.

7.3.3 Optimising the Model Recursively

The advantage of using seed triplets for model fitting is that a seed triplet has a higher probability of containing only true positives than a sample set that is drawn randomly from all candidates in the sliding window [44]. However, even if a seed triplet is free of false positives, the model computed with it is usually poor, as estimates are given by extrapolation. This can be seen in

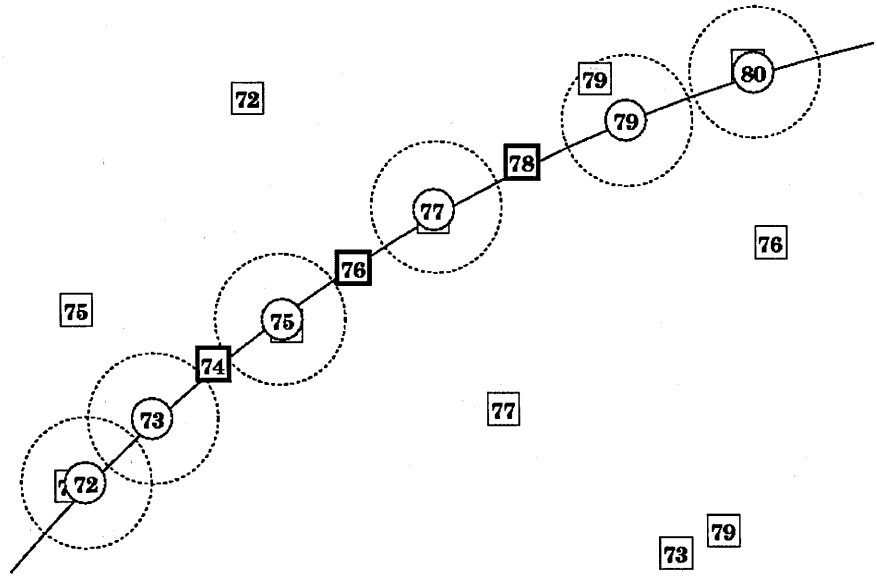


Figure 7.4: Optimising the fitted model. Assume in Fig. 7.3.2 two candidates, one from $\mathcal{C}_{74}$ and one from $\mathcal{C}_{78}$ are consistent with the model fitted to the seed triplet, *i.e.* $\dot{k} = 74$, $\dddot{k} = 78$, and $\ddot{k} = 76$. The new triplet (bold squares in (b)) is then used to compute a "better" model.

Fig. 7.3.2. As the estimates get further away from the seed triplet on the time axis, they depart from the detected ball positions in row-column plane rapidly.

We remedy this problem by recursively optimising the model using supports found in the previous iteration [9]. First, we define the support of a model to be a candidate that is consistent with the model. A candidate $c_k^j \in \mathcal{C}^{(i)}$ located at $\mathbf{z}_k^j$ is said to be a support if

$$d(\hat{\mathbf{z}}_k, \mathbf{z}_k^j) < d_{th} \tag{7.4}$$

where $d(\cdot,\cdot)$ is the Euclidean distance between two points; $\hat{\mathbf{z}}_k$ is the estimated ball position at time k as given by the model; d_{th} is a predefined threshold. In the rare case where more than one candidate satisfies (7.4) at time k, only the one with the smallest distance is selected as a support.

Let $\mathcal{S}$ be the set of supports of a model. Also let

$$\begin{aligned} \dot{k} &\triangleq \min\ k \quad \forall c_k^j \in \mathcal{S}, \qquad \dddot{k} \triangleq \max\ k \quad \forall c_k^j \in \mathcal{S}, \\ \ddot{k} &\triangleq \arg\min_k \left|\left|\, |\dddot{k}-k| - |k-\dot{k}| \,\right|\right| \quad \forall c_k^j \in \mathcal{S} \end{aligned} \tag{7.5}$$

Now we use the 3 candidates in $\mathcal{S}$ from frame $\dot{k}$, $\ddot{k}$ and $\dddot{k}$ as a new triplet to fit another model. Since the candidates in the new triplet are further apart from each other, more estimates are

interpolated. The resulting model is usually "better" than the one computed with the seed triplet. One iteration of the optimisation is thus complete (Fig. 7.3.3).

7.3.4 Stopping Criteria

Now we need a measure of the quality of a model. RANSAC-like algorithms normally use the number of supports a model gets. In [67], a maximum likelihood version of RANSAC, MLESAC, is proposed to give a more accurate measure. However, MLESAC involves estimation of mixture parameters of a likelihood function [36, 65], which can be complicated and computationally expensive. In our implementation, the following cost function [67] is adopted:

$$C = \sum_{k=i-V}^{i+V} \sum_{j} \rho(\mathbf{z}_k^j) \tag{7.6}$$

where

$$\rho(\mathbf{z}_k^j) = \begin{cases} d^2(\hat{\mathbf{z}}_k, \mathbf{z}_k^j) & \text{if } d(\hat{\mathbf{z}}_k, \mathbf{z}_k^j) < d_{th} \\ d_{th}^2 & \text{if } d(\hat{\mathbf{z}}_k, \mathbf{z}_k^j) \geq d_{th} \end{cases} \tag{7.7}$$

and a smaller C indicates a better model. This cost function provides a more accurate measure of a model's quality than "number of supports". This is achieved by assigning different weights to supports according to their "fitness" to the model.

Having defined C, the optimisation loop terminates when the support set $\mathcal{S}$ stops expanding, or when C starts to increase. More specifically, let $M^{(b)}$ be the model after the b^{th} iteration, $C^{(b)}$ be its cost, $\dot{k}^{(b)}$ and $\dddot{k}^{(b)}$ are defined as in (7.5). The optimisation loop terminates if

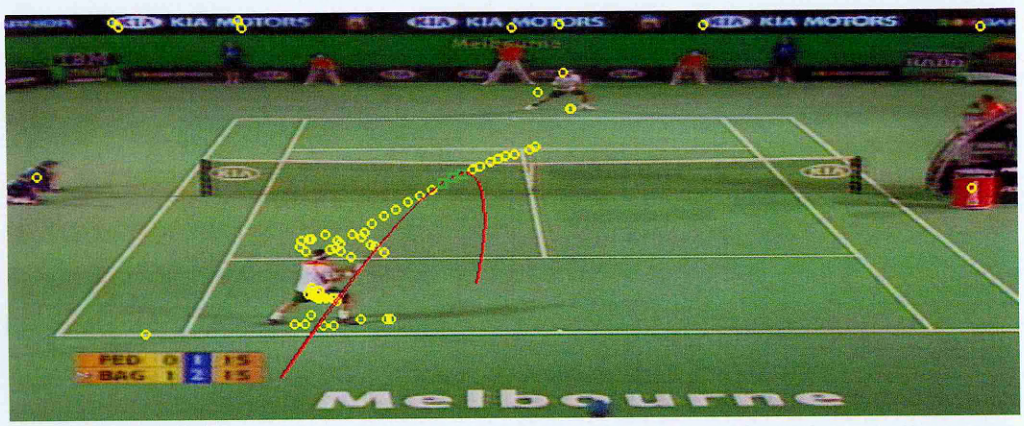
$$\dot{k}^{(b+1)} = \dot{k}^{(b)} \quad \wedge \quad \dddot{k}^{(b+1)} = \dddot{k}^{(b)} \tag{7.8}$$

or

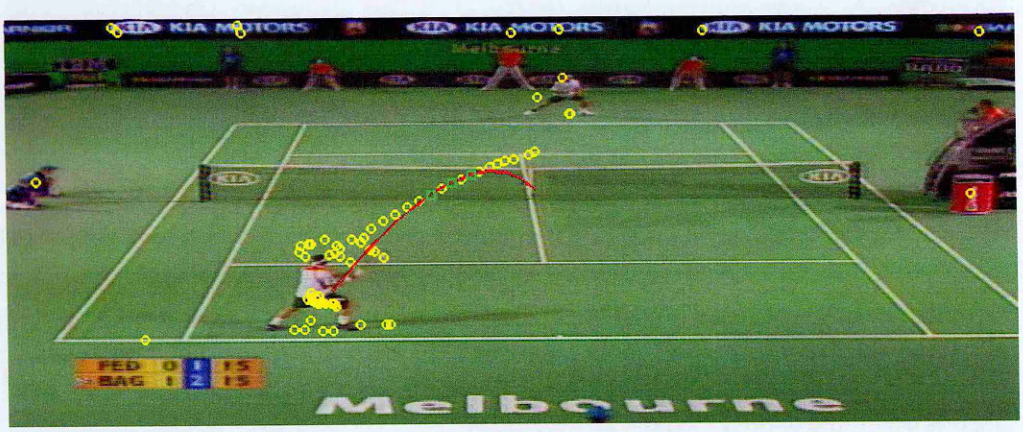
$$C^{(b+1)} > C^{(b)} \tag{7.9}$$

The latter case happens when a clutter-originated candidate is included in a triplet for model fitting, or when $\mathcal{S}$ is extended in the time domain, and the departure of the constant acceleration model from measurements becomes significant.

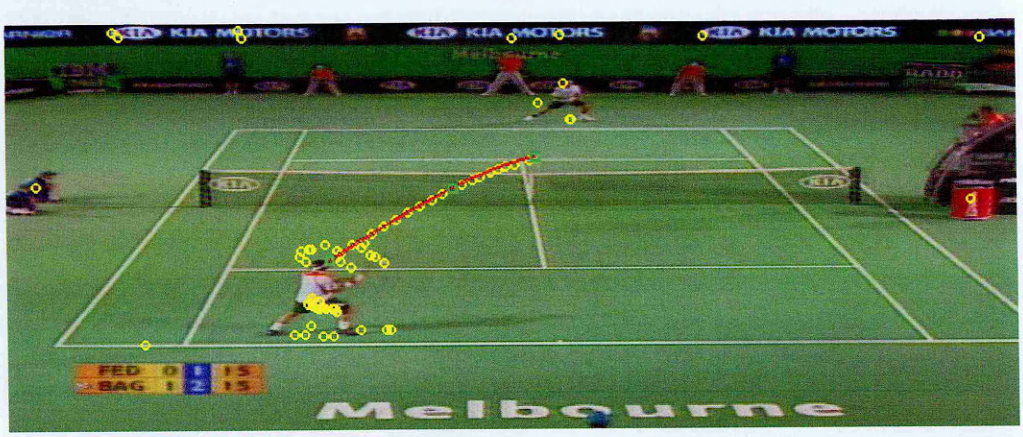
Once the optimisation is complete, $M^{(b)}$ is retained as the final fitted model to the current seed triplet. If $M^{(b)}$ is ball-originated, it is now usually well converged to the "true motion" of the ball (see Fig. 7.3.4).



(a)



(b)



(c)

Figure 7.5: Model fitting and model optimisation. Yellow circles: ball candidates inside an interval of 31 frames ($V = 15$). Green squares: candidates that were used for model fitting. Red line: fitted dynamic model. (a) Fitting a dynamic model to the seed triplet. (b) and (c): the first and the third iteration of model optimisation. Convergence is achieved after the third iteration.

7.3.5 Defining Tracklet

Now we introduce the concept of **tracklet**. A tracklet T is defined as the combination of a parameterised model M and its support set $\mathcal{S}$, *i.e.* $T \triangleq \{M, \mathcal{S}\}$. According to this definition, what we finally get from a seed triplet is a tracklet with an optimised motion model and its support set. For interval I_i, the above model-fitting/model-optimising process is applied to each seed triplet that contains each c_i^j in $\mathcal{C}_i$. We then threshold all the generated tracklets using the number of supports. Only the tracklets that have more than m_{th} supports are retained, and the l^{th} retained tracklet in I_i is denoted by $T_i^l = \{M_i^l, \mathcal{S}_i^l\}$.

As the sliding window moves, a sequence of tracklets is generated. Fig. 7.3.5 plots all 672 tracklets generated in the example sequence in the row-column-time 3D space. The candidates in this sequence were shown in Fig 7.2. In Fig. 7.3.5, a tracklet T_i^l is plotted as its dynamic model M_i^l, and is bounded by $\dot{k}_i^l$ and $\dddot{k}_i^l$.

Comparing Fig 7.2 and Fig. 7.3.5 (a), we can see that the latter is much "cleaner". This is because clutter-originated candidates that are not persistent are unlikely to form seed triplets, and thus unlikely to form tracklets. From a signal processing point of view, the "signal" (true candidate/tracklet) to "noise" (false candidate/tracklet) ratio in the tracklet space is much higher than that in the candidate space. This implies that performing association in the tracklet space should be easier than in the candidate space. Moreover, the complexity of the association problem is reduced, since there are far fewer tracklets than candidates (in this example 672 and 6749 respectively).

7.4 Tracklet Level Association

The generated tracklets may have originated from different balls or from clutter. Now a method for tracklet level data association is needed. In this section, we formulate tracklet level association as a single-pair shortest path (SPSP) problem. First, we construct a weighted and directed graph where each node is a tracklet, and the edge weight between two nodes is defined according to the "compatibility" of the two tracklets. Assuming for the moment that there is only one ball in the sequence, and that the first and last tracklets (nodes) that have originated from this ball

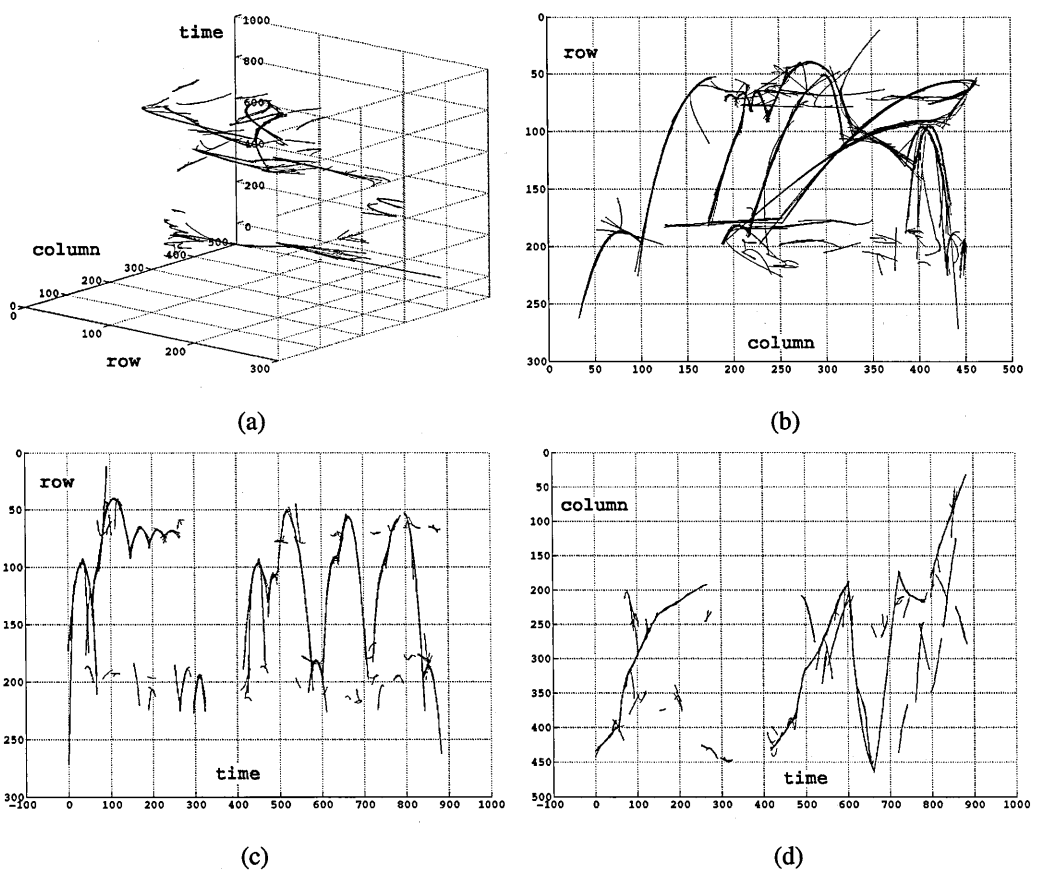


Figure 7.6: All the generated tracklets in the example sequence. (a) 3D view. (b) Projection on the row-column plane. (c) Projection on the row-time plane. (d) Projection on the column-time plane.

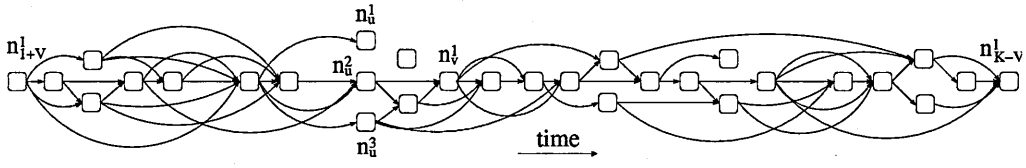


Figure 7.7: An illustrative example of the graph topology. Nodes in the same stage are aligned vertically. Note that the number of nodes in a typical sequence is of the order of 10^2 to 10^3. Far fewer nodes are plotted in this figure for ease of visualisation.

are already known, we show that the ball trajectory can be identified by looking for the shortest path between the two nodes in the graph.

7.4.1 Constructing a Weighted and Directed Graph

The tracklet level association problem is mapped onto a graph. We construct a weighted and directed graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ is the set of nodes, and $\mathcal{E}$ is the set of edges. Each node in $\mathcal{N}$ is a tracklet, and the node corresponding to tracklet T_i^l is denoted by n_i^l. Clearly, $\mathcal{N}$ is composed of stages as

$$\mathcal{N} = \{\mathcal{N}_{1+V}, \mathcal{N}_{2+V}, ..., \mathcal{N}_{K-V-1}, \mathcal{N}_{K-V}\} \tag{7.10}$$

where the i^{th} stage $\mathcal{N}_i$ is the set of nodes (tracklets) generated in interval I_i; I_i centres on frame i and spans frame $i - V$ to frame $i + V$; and K is the number of frames in the sequence. A directed edge from node T_u^p to node T_v^q, $e_{u,v}^{p,q}$, exists in $\mathcal{E}$, if

$$u < v \quad \wedge \quad \dot{k}_v^q - \ddot{k}_u^p \leq k_{th} \tag{7.11}$$

where $\dot{k}_v^q$ is the $\dot{k}$ of tracklet T_v^q, $\ddot{k}_u^p$ is the $\ddot{k}$ of tracklet T_u^p, and $u, v \in [1 + V, K - V]$. The assumption here is that misdetection of the ball can happen in at most k_{th} successive frames. An example of graph topology is given in Fig. 7.4.1, where nodes in the same stage are aligned vertically.

Both M and $\mathcal{S}$ of the tracklets are used to define the edge weight between two nodes. First, two nodes n_u^p and n_v^q are said to be "overlapping" if

$$u < v \quad \wedge \quad \dot{k}_v^q - \ddot{k}_u^p \leq 0 \tag{7.12}$$

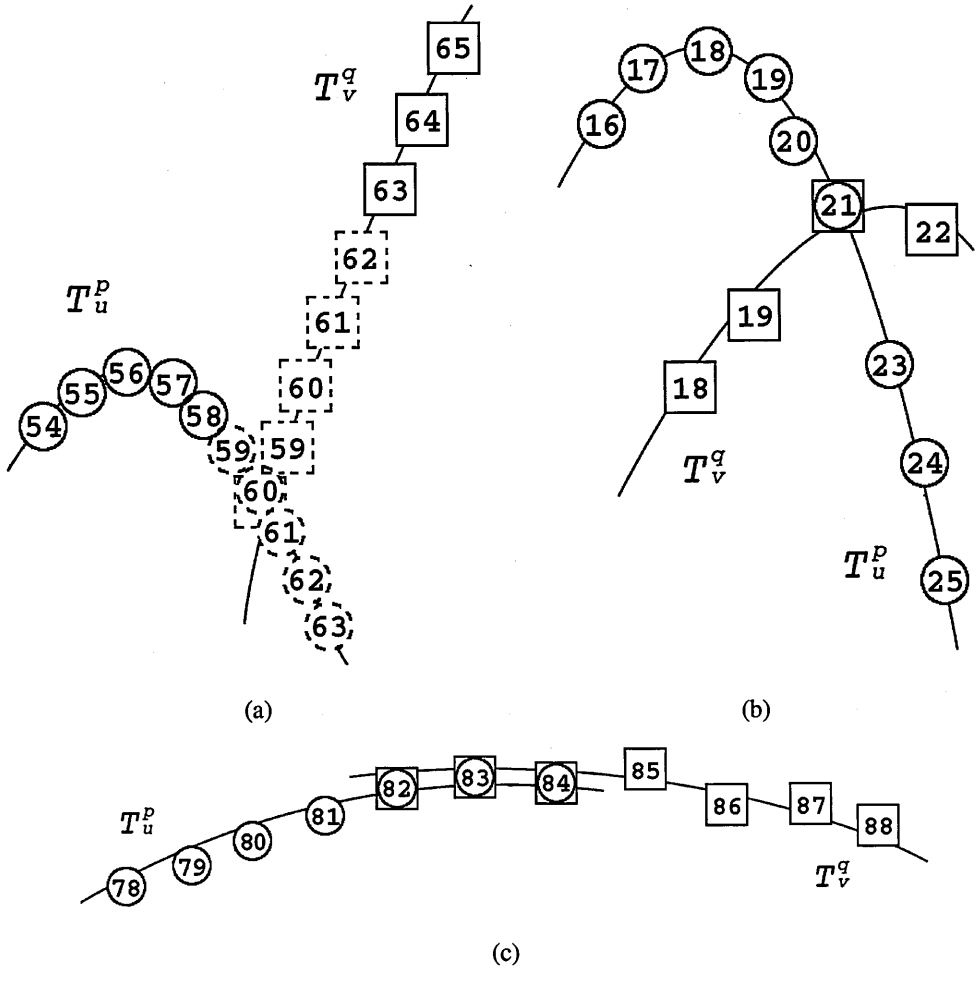


Figure 7.8: Edge weight between two nodes. Solid circles and squares: supports of T_u^p and T_v^q respectively. Dashed circles and squares: estimates given by M_u^p and M_v^q respectively. (a) Non-overlapping case. $w_{u,v}^{p,q} = d(\hat{\mathbf{z}}_{u,59}^p, \hat{\mathbf{z}}_{v,59}^q)$. (b) Overlapping and conflicting case. $w_{u,v}^{p,q} = \infty$. (c) Overlapping and non-conflicting case. $w_{u,v}^{p,q} = 0$.

For overlapping nodes, their support sets are used for defining the edge weight between them. Two overlapping nodes are "conflicting" (not compatible) if for $\dot{k}_v^q \leq k \leq \dddot{k}_u^p$, $\exists k$ such that

$$(\exists c_k^{j'} \in \mathcal{S}_u^p \ \wedge \ \nexists c_k^{j''} \in \mathcal{S}_v^q) \quad \vee \quad (\nexists c_k^{j'} \in \mathcal{S}_u^p \ \wedge \ \exists c_k^{j''} \in \mathcal{S}_v^q) \tag{7.13}$$
$$\vee \quad (\exists c_k^{j'} \in \mathcal{S}_u^p \ \wedge \ \exists c_k^{j''} \in \mathcal{S}_v^q \ \wedge \ \mathbf{z}_k^{j'} \neq \mathbf{z}_k^{j''})$$

In other words, two compatible tracklets should agree on the support in every frame in the overlapping region: either both having the same support, or both having no support. The edge weight between two overlapping nodes is then given by

$$w_{u,v}^{p,q} = \begin{cases} \infty & \text{if } n_u^p \text{ and } n_v^q \text{ are conflicting} \\ 0 & \text{otherwise} \end{cases} \tag{7.14}$$

For non-overlapping nodes, their parameterised models are used. Ball positions from frame $\dddot{k}_u^p$ to frame $\dot{k}_v^q$ are estimated. The edge weight is then defined as

$$w_{u,v}^{p,q} = \min d(\hat{\mathbf{z}}_{u,k}^p, \hat{\mathbf{z}}_{v,k}^q), \ \ \forall k \in [\dddot{k}_u^p, \dot{k}_v^q] \tag{7.15}$$

7.4.2 Finding the Single-Pair Shortest Path

A path in a directed graph is a sequence of nodes such that from each node (except the last one) there is an edge going to the next node in the sequence. The first node in the sequence is called the source node and the last node is called the destination node. In an edge-weighted graph, the weight of a path is defined as the sum of the weights of all the edges it goes through. The shortest path between two nodes is the path with the smallest weight among all possible paths.

The edge weight defined in (7.14) and (7.15) can be thought of as a compatibility measure: the smaller the edge weight between two nodes is, the more likely the two tracklets have originated from the same ball. For the moment, we assume that there is only one ball to track, and that the first and last nodes that have originated from this ball are already known. We show that the ball trajectory can be identified by looking for the shortest path between the two nodes in the graph. In the next section, we will relax the assumptions made above by introducing all-pairs shortest path and path level analysis.

Dijkstra's algorithm is a single-source multi-destination shortest path algorithm for a directed graph with non-negative edge weights [19], which, of course, can be used to solve our single-pair shortest path problem. We apply Dijkstra's algorithm to the example sequence we have been

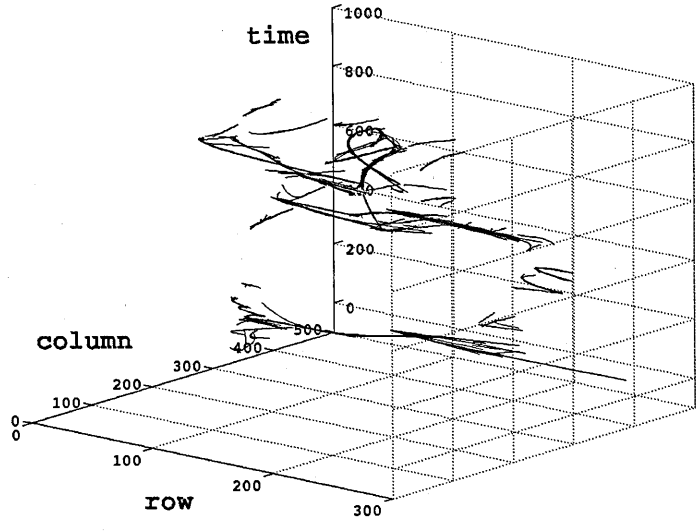


Figure 7.9: All generated tracklets in the example sequence. The first and last nodes in the third play are plotted as red thick lines. Dijkstra's algorithm is to be applied to find the shortest path between these two nodes.

using in the previous sections. We manually specify the first and last ball-originated nodes in the third play in the sequence as source node and destination node (see Fig. 7.4.2), respectively, and use Dijkstra's algorithm to find the shortest path between the two nodes.

Fig. 7.4.2 shows the result of applying Dijkstra's algorithm: a shortest path that does correspond to the ball trajectory. The shortest path contains 26 nodes and goes through 25 edges. The overall weight of the shortest path is 7.1 pixels. According to the definition of edge weight in (7.14) and (7.15), the shortest path found is guaranteed to be non-conflicting: at any time k, there is at most one candidate in the support sets of the shortest path. The data association problem is thus solved. The reconstructed ball trajectory after interpolation and event detection is shown in Fig. 7.4.2. An illustration of the shortest path in the graph is shown in Fig. 7.4.2.

7.5 Path Level Analysis

In this section, we relax the assumptions made in the previous section, and show how this can lead to a fully automatic algorithm for tracking multiple balls. The key idea is to use all-pairs shortest path (APSP) instead of single-pair shortest path (SPSP) at the tracklet level, and introduce one more layer on top of that, namely, path level analysis. Note that the choice of APSP

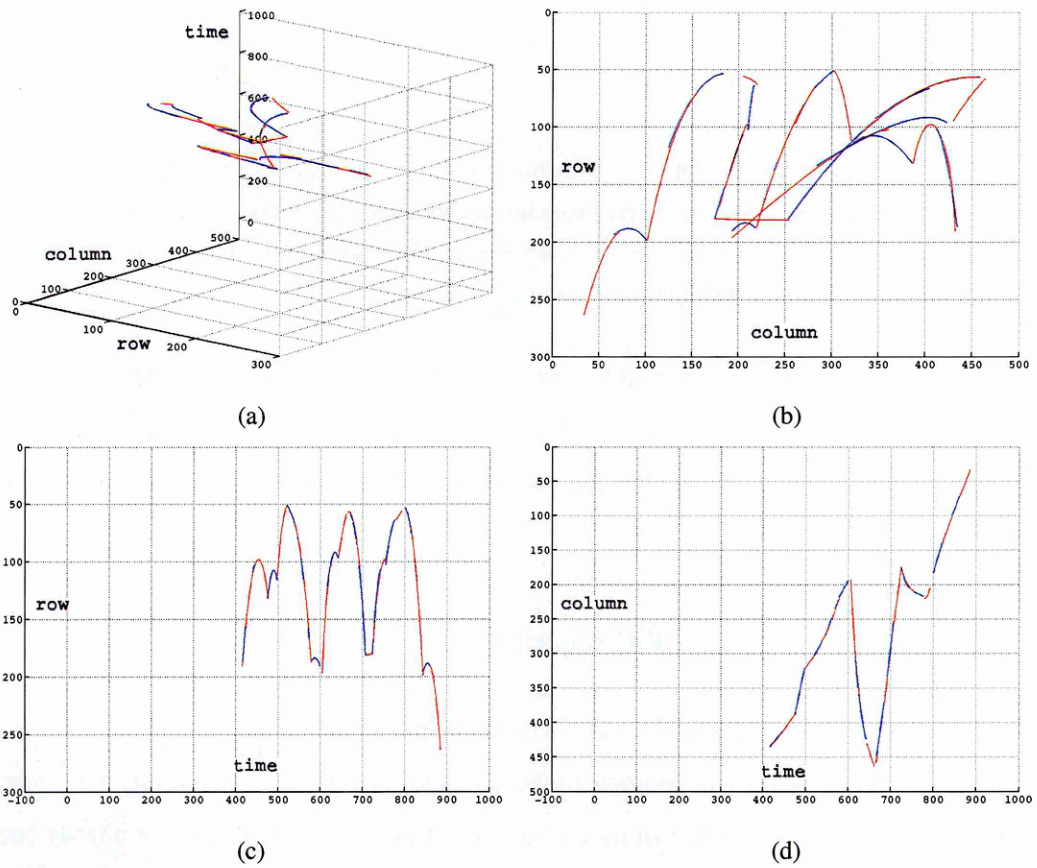


Figure 7.10: Shortest path given by Dijkstra's algorithm. Adjacent nodes in the shortest path are plotted alternatively in blue and red. (a) 3D view. (b) Projection on the row-column plane. (c) Projection on the row-time plane. (d) Projection on the column-time plane.

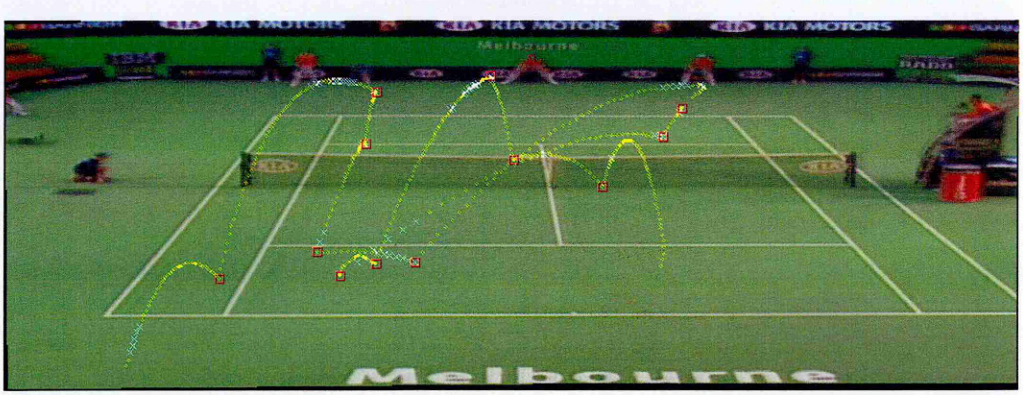


Figure 7.11: Ball trajectory superimposed on the mosaic image. Yellow circles: positions of the candidates in the support sets of the nodes in the shortest path. White crosses: interpolated tennis ball positions. Red squares: detected key events.

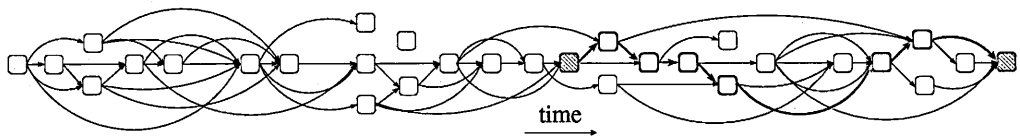


Figure 7.12: An illustration of the shortest path in Fig. 7.4.2 in the graph. Striped red squares: the manually specified source node and destination node. Red squares: the shortest path between the source node and the destination node. The number of nodes in a typical sequence is of the order of 10^2 to 10^3. Far fewer nodes are plotted in this figure for ease of visualisation.

algorithm is not important. In principle, any APSP algorithm is applicable. However, in order to make the complete data association algorithm more efficient, we have developed a fast APSP algorithm. We will present it in the next section, under the context of processing speed considerations.

7.5.1 The Paths Reduction (PR) Algorithm

For a given pair of nodes n_u^p and n_v^q in $\mathcal{G}$, there may be paths connecting n_u^p to n_v^q, or there may not be any such path at all. One example of the latter case is when $u \geq v$. Assume the shortest paths between all pairs of nodes that have at least one path connecting them have already been identified. Let $\mathcal{Q}$ be the set of such all-pairs shortest paths, and Q is the number of paths in $\mathcal{Q}$. Q is in the order of N^2, where N is the number of nodes in the graph. Now observe that no matter how many balls there are to track, or where each of the ball trajectories starts and terminates, according to the previous section, the paths that correspond to the ball trajectories form a subset of $\mathcal{Q}$. The question now is how to reduce the original set of APSP $\mathcal{Q}$ to its subset that contains only paths that correspond to the ball trajectories.

For now let us assume that we can order the Q paths in $\mathcal{Q}$ according to their "qualities", and that we know the pair-wise "compatibility" of the paths, either compatible or incompatible. We will justify these two assumptions shortly. Having these two assumptions, we define a subset of $\mathcal{Q}$ generated by a paths reduction (PR) algorithm (see Algorithm 1) to be the best set of compatible paths (BSCP), and denote it by $\mathcal{B}$.

We illustrate this algorithm using a synthesised example. Let $\mathcal{Q} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$, where P_1 to P_6 are 6 paths. Also let $P_1 > P_2 > P_3 > P_4 > P_5 > P_6$, where ">" denotes "is better than". The pair-wise compatibility is as in Table 7.5.1, where "$\odot$" denotes "compatible",

Table 7.1: The Paths Reduction (PR) Algorithm

Initialisation: $\mathcal{Q}$ has Q paths, and $\mathcal{B}$ is empty.

While $\mathcal{Q}$ is not empty:

- Remove the best path P^* in $\mathcal{Q}$ from $\mathcal{Q}$;

- If P^* is compatible with all paths in $\mathcal{B}$, add P^* to $\mathcal{B}$.

Table 7.2: Pair-wise compatibility of the 6 paths in $\mathcal{Q}$

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	-	$\otimes$	$\odot$	$\odot$	$\otimes$	$\odot$
P_2	$\otimes$	-	$\otimes$	$\odot$	$\otimes$	$\otimes$
P_3	$\odot$	$\otimes$	-	$\otimes$	$\otimes$	$\odot$
P_4	$\odot$	$\odot$	$\otimes$	-	$\otimes$	$\odot$
P_5	$\otimes$	$\otimes$	$\otimes$	$\otimes$	-	$\otimes$
P_6	$\odot$	$\otimes$	$\odot$	$\odot$	$\otimes$	-

and "$\otimes$" denotes "incompatible". Since there are 6 paths in the original set $\mathcal{Q}$, the BSCP $\mathcal{B}$ is obtained after 6 iterations. Paths in $\mathcal{Q}$ and $\mathcal{B}$ after each iteration are shown in Table 7.5.1.

7.5.2 Defining Quality and Compatibility of Paths

Now we define the quality and compatibility of paths. First, recall that the weight of a path is the sum of the weights of all edges the path goes through. We define the length of a path to be the number of supports in all its nodes, or more precisely, the size of the union of the support sets in all its nodes. It should be noted that according to the definitions of weight and length of a path, the term "shortest path" used in the previous sections should have been "lightest path".

Table 7.3: Paths in $\mathcal{Q}$ and $\mathcal{B}$ after each iteration of the PR algorithm.

iteration	$\mathcal{Q}$	$\mathcal{B}$
0	P_1,P_2,P_3,P_4,P_5,P_6	$\emptyset$
1	P_2,P_3,P_4,P_5,P_6	P_1
2	P_3,P_4,P_5,P_6	P_1
3	P_4,P_5,P_6	P_1,P_3
4	P_5,P_6	P_1,P_3
5	P_6	P_1,P_3
6	$\emptyset$	P_1,P_3,P_6

However, we chose to use "shortest path" for the sake of consistency with the terminology used in other papers.

Intuitively, a "good" path is one that is both "light" and "long". However, there is usually a trade-off between the weight and the length of a path: a longer path tends to be heavier. Taking this into account, we define the relative quality of two path P_1 and P_2 as follows:

$$P_1 \left\{ \begin{array}{c} > \\ = \\ < \end{array} \right\} P_2 \quad \text{if} \quad (W_1 - W_2) \left\{ \begin{array}{c} < \\ = \\ > \end{array} \right\} \alpha \cdot (L_1 - L_2) \tag{7.16}$$

where the relation operators ">", "=" and "<" between P_1 and P_2 stand for "is better than", "has the same quality as", and "is worse than", respectively; W_1 and W_2 are the weights of P_1 and P_2, respectively; L_1 and L_2 are the lengths of P_1 and P_2, respectively; and α is a controllable parameter with the unit of pixel. According to this definition, if a path P_1 is "much longer" but "slightly heavier" than a path P_2, then P_1 is said to have a better quality than P_2. Note that this definition does not assume any relationship between W_1 and W_2, or relationship between L_1 and L_2.

It easily follows that the set $\mathcal{Q}$ equipped with an operator "$\geq$" satisfies the following three

statements:

1. **Transitivity:** if $P_1 \geq P_2$ and $P_2 \geq P_3$ then $P_1 \geq P_3$;
2. **Antisymmetry:** if $P_1 \geq P_2$ and $P_2 \geq P_1$ then $P_1 = P_2$;
3. **Totality:** $P_1 \geq P_2$ or $P_2 \geq P_1$.

According to order theory [13], $\mathcal{Q}$ associated with operator "$\geq$" is a totally ordered set. The first assumption for the PR algorithm to work is satisfied.

The second assumption, the existence of pair-wise compatibility of the paths, is straightforward. Two paths are said to be compatible if and only if they do not share any common **support**. It should be noted, however, two paths that do not share any common **node** are not necessarily compatible, because different nodes can have common supports.

7.5.3 Applying the PR Algorithm to $\mathcal{Q}$

Having ordered the paths in $\mathcal{Q}$ and obtained their pair-wise compatibilities, we can now apply the PR algorithm to $\mathcal{Q}$. This means we finally have a complete and fully automatic data association algorithm.

We apply the complete algorithm to the example sequence. In Section 7.4.2, we manually specified the first and the last true nodes in the third play in the sequence, and used SPSP to find the shortest path between them. Now, with path level analysis, track initiation and track termination is automated, and multiple plays can be handled. By looking for all-pairs shortest paths, a set $\mathcal{Q}$ with $Q = 87961$ paths is obtained. The PR algorithm is then applied, which gives a BSCP $\mathcal{B}$ containing 11 paths. In descending order, the numbers of supports (lengths) of the paths in $\mathcal{B}$ are: 411, 247, 62, 23, 20, 17, 17, 16, 15, 10, 9. It is a reasonable assumption that a path corresponding to a ball trajectory has more supports than a path corresponding to the motion of a non-ball object, *e.g.* a wristband. We set a threshold L_{th} and keep only the paths that have more supports than L_{th}. This results in a thresholded BSCP $\mathcal{B}_{th}$ with 3 paths, where each path corresponds to a play in the sequence. The paths in $\mathcal{B}_{th}$ are plotted in the 3D space in Fig. 7.5.3. In Fig. 7.5.3, the 3 ball trajectories are superimposed on mosaic images. An illustration of the 3 paths in the graph is shown in Fig. 7.5.3.

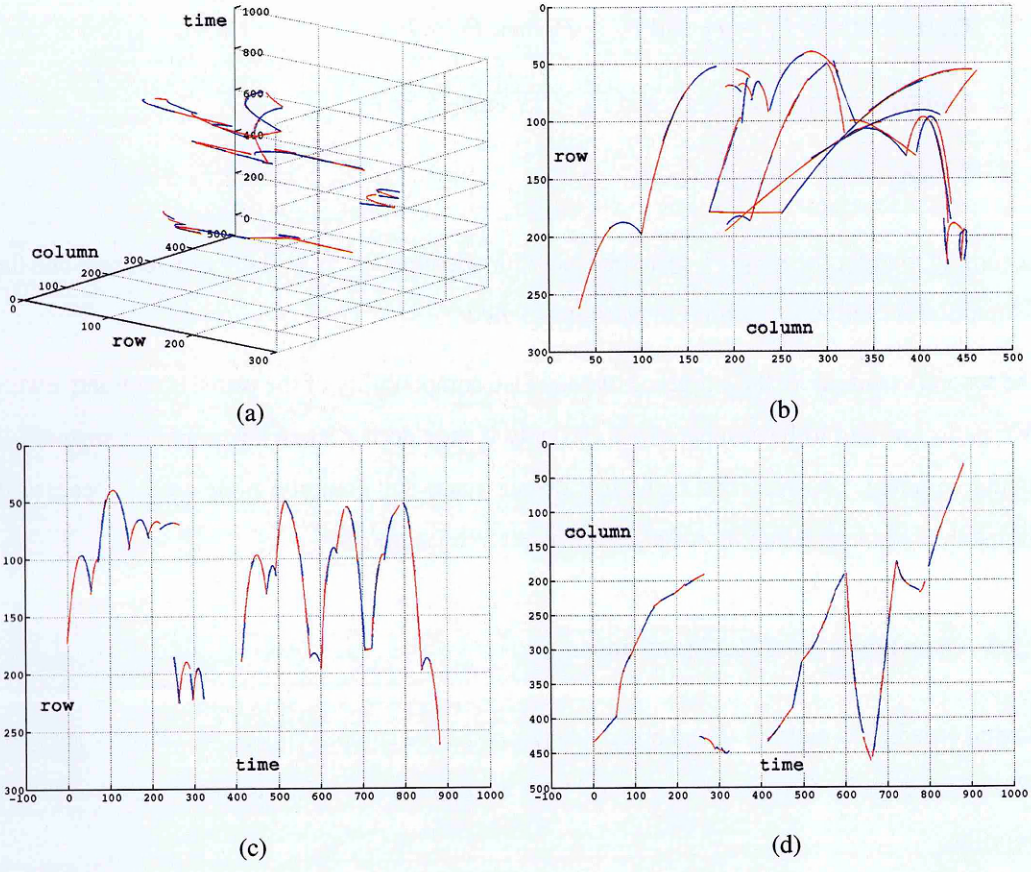


Figure 7.13: Results of applying the complete algorithm to the example sequence: 3 paths in the thresholded BSCP $\mathcal{B}_{th}$. Adjacent nodes in each path are plotted alternatively in blue and red. (a) 3D view. (b) Projection on the row-column plane. (c) Projection on the row-time plane. (d) Projection on the column-time plane.

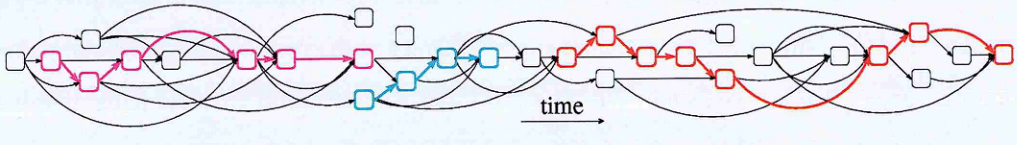
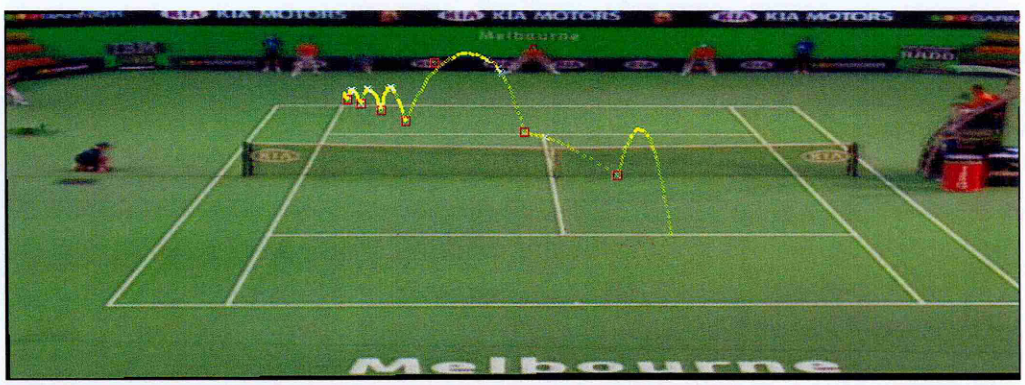
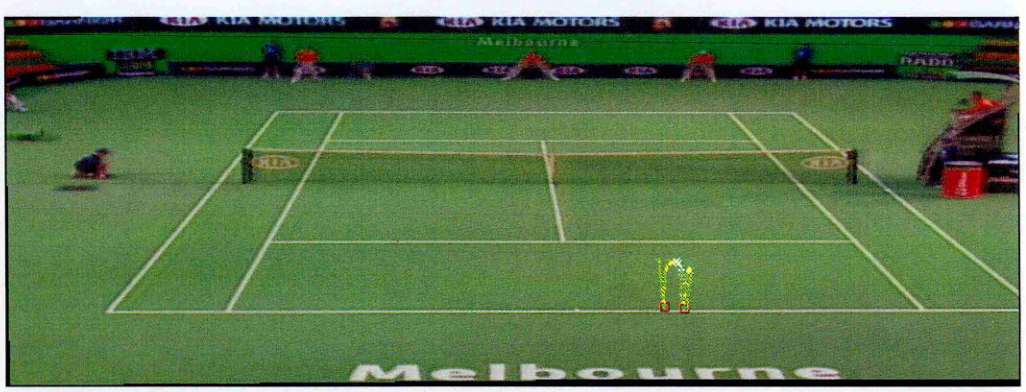


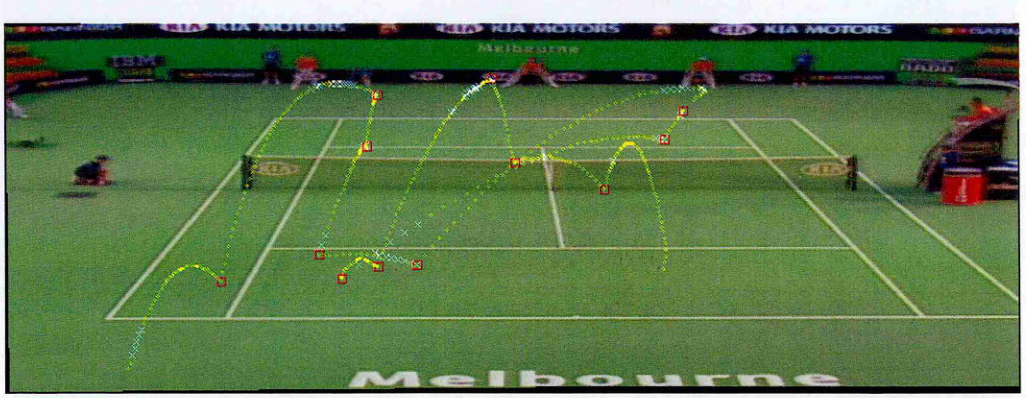
Figure 7.14: An illustration of the paths in $\mathcal{B}_{th}$. Magenta, cyan, and red paths correspond to the first, second and third play in the sequence, respectively.



(a)



(b)



(c)

Figure 7.15: Ball trajectories (after interpolation and key event detection) superimposed on mosaic images. From (a) to (c): the first, second and third play in time order. The first and second plays overlap in time: the first play spans frame 16 to frame 260; the second frame 254 to frame 321; and the third frame 427 to frame 887.

7.6 Processing Speed Considerations

This section discusses processing speed considerations pertinent to a practical implementation of the proposed algorithm. We will discuss the complexity of each layer of the complete algorithm, and show that by applying various speeding-up techniques, the processing time can be kept reasonable even on very large problems. In particular, by exploiting a special topological property of the graph, we develop an APSP algorithm which has lower complexity than the general-purpose ones.

7.6.1 Candidate Level Association

Because of the way the tracklets are generated, the complexity of candidate level association is linear in the number of frames in the sequence. When there are more than 3 candidates inside the ellipsoid around a candidate, all possible combinations that can form seed triplets are considered. However, since the size of the ellipsoid is very small, in practice this happens very rarely. As a result, the complexity of candidate level association is approximately linear in the number of candidates in each frame.

7.6.2 Tracklet Level Association

At the tracklet level, we need to solve an APSP problem for a graph $\mathcal{G}$ with N nodes. N is normally of the same order as the number of frames in the sequence, which ranges from less than a hundred to more than one thousand. Obviously, Dijkstra's algorithm could be applied repeatedly to solve an APSP problem. However, the efficiency of such a scheme would be low. Several dedicated APSP algorithms have been proposed. The Floyd-Warshall algorithm solves APSP on a directed graph in $O(N^3)$ time [22]. Johnson's algorithm, taking advantage of the sparsity of a graph, has a complexity of $O(N^2 \log N + NE)$, where E is the number of edges in the graph [29].

Neither the Floyd-Warshall algorithm nor Johnson's algorithm makes any assumption about the topology of the graph. According to the way our graph is constructed, its topology has a special property. By exploiting this special property, we have developed a more efficient APSP algorithm than the general-purpose ones.

First, let us recall the concept of a trellis. A trellis is a special type of directed graph, where the set of nodes $\mathcal{N}$ can be partitioned into ordered subsets (stages) $\mathcal{N}_1, \mathcal{N}_2, ...$, such that edges exist only from nodes in stage $\mathcal{N}_{i-1}$ to nodes in stage $\mathcal{N}_i$. The graph $\mathcal{G}$ generated in Section 7.4.1 is not a trellis. However, it satisfies a weaker condition. According to (7.10) and (7.11), $\mathcal{N}$ can be partitioned into stages $\mathcal{N}_{1+V}, \mathcal{N}_{2+V}, ..., \mathcal{N}_{K-V-1}, \mathcal{N}_{K-V}$, such that edges exist only from nodes in stage $\mathcal{N}_u$ to nodes in stage $\mathcal{N}_v$ if $u < v$. We call a graph with this property a **semi-trellis**. Using the fact that $\mathcal{G}$ is a semi-trellis, we derive an $O(N^2)$ APSP algorithm as follows.

The proposed APSP algorithm uses the concept of dynamic programming. Assume we have a sub-graph of the complete graph $\mathcal{G}$, and APSP in this sub-graph has been solved. As more tracklets are generated, new nodes and edges are added to the sub-graph. APSP in this augmented sub-graph is then solved using the solution of APSP in the previous sub-graph. When the complete graph $\mathcal{G}$ is constructed, the APSP problem in it is solved.

The proposed APSP algorithm runs in the same pass as tracklet generation. Suppose we are in the middle of the tracklet generation process. The sliding window now centres on frame $i-1$, and tracklets in interval I_{i-1} have been generated. Let $\mathcal{G}^{(i-1)} = \{\mathcal{N}^{(i-1)}, \mathcal{E}^{(i-1)}\}$ be the graph constructed so far, where $\mathcal{N}^{(i-1)} = \{\mathcal{N}_{1+V}, \mathcal{N}_{2+V}, ..., \mathcal{N}_{i-1}\}$ is the set of nodes from stages $1+V$ to $i-1$; $\mathcal{E}^{(i-1)}$ is the set of edges that go into all nodes in $\mathcal{N}^{(i-1)}$. Clearly, $\mathcal{G}^{(i-1)}$ is a sub-graph of the complete graph $\mathcal{G}$. Assume the APSP problem in graph $\mathcal{G}^{(i-1)}$ has been solved. In each node $n_v^q \in \mathcal{G}^{(i-1)}$, a table is maintained, where each entry corresponds to a node in the sub-graph $\mathcal{G}^{(v-1)}$. The entry corresponding to node $n_u^p \in \mathcal{G}^{(v-1)}$ keeps two pieces of information about the shortest path from n_u^p to n_v^q in $\mathcal{G}^{(i-1)}$. The first piece of information is the last node before n_v^q in the shortest path from n_u^p to n_v^q, and the second piece of information is the total weight of the shortest path from n_u^p to n_v^q. It should be noted that in the table in node n_v^q, no information is kept about the shortest path from a node n_u^p to n_v^q if $u \geq v$. This is because $\mathcal{G}^{(i-1)}$ is a semi-trellis: there is no path going from n_u^p to n_v^q if $u \geq v$.

Next, we show how to solve the APSP problem in $\mathcal{G}^{(i)}$ using the solution of the APSP problem in $\mathcal{G}^{(i-1)}$. Now the sliding window moves one frame forward, and the interval I_i centres on frame i. Assume several tracklets are generated in I_i, forming the set of nodes $\mathcal{N}_i$. Now we need to construct for each node $n_i^l \in \mathcal{N}_i$ a table of APSP knowledge, where each entry contains information about the shortest path in $\mathcal{G}^{(i)}$ from a node in $\mathcal{G}^{(i-1)}$ to n_i^l. In Fig. 7.6.2, the sub-

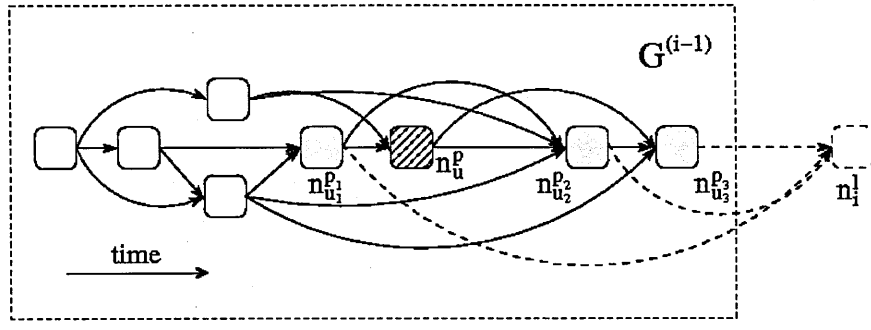


Figure 7.16: Constructing the table of APSP knowledge for a node $n_i^l \in \mathcal{N}_i$. The sub-graph in the big rectangle: $\mathcal{G}^{(i-1)}$. Dashed node: a node $n_i^l \in \mathcal{N}_i$. Shaded nodes: nodes in $\mathcal{G}^{(i-1)}$ that are connected to n_i^l with edges. Striped node: an arbitrary node $n_u^p \in \mathcal{G}^{(i-1)}$.

graph inside the big rectangle represents $\mathcal{G}^{(i-1)}$, and a new node $n_i^l \in \mathcal{N}_i$ is plotted as a dashed node. Assume s nodes in $\mathcal{G}^{(i-1)}$ are connected to n_i^l with edges. These s nodes are denoted by $n_{u1}^{p1}, n_{u2}^{p2}, ..., n_{us}^{ps}$, and are plotted as shaded nodes in Fig. 7.6.2. Edges that connect these nodes to n_i^l are denoted by $e_{u1,i}^{p1,l}, e_{u2,i}^{p2,l}, ..., e_{us,i}^{ps,l}$, and plotted as dashed edges. Obviously, the number of entries in the table of APSP knowledge in n_i^l is equal to the number of nodes in $\mathcal{G}^{(i-1)}$. Without loss of generality, let us consider one entry in the table, which keeps information about the shortest path in $\mathcal{G}^{(i)}$ from a node $n_u^p \in \mathcal{G}^{(i-1)}$ to n_i^l. In Fig. 7.6.2, n_u^p is plotted as a striped node. Now observe that the shortest path from n_u^p to n_i^l in $\mathcal{G}^{(i)}$ must go through one of the nodes in $n_{u1}^{p1}, n_{u2}^{p2}, ..., n_{us}^{ps}$ and the corresponding edge in $e_{u1,i}^{p1,l}, e_{u2,i}^{p2,l}, ..., e_{us,i}^{ps,l}$. Since APSP has been solved in $\mathcal{G}^{(i-1)}$, the information about the shortest path in $\mathcal{G}^{(i-1)}$ from n_u^p to n_{ur}^{pr} is kept in the table in n_{ur}^{pr}, where $r = 1, 2, ..., s$. Let $W^{(i-1)}(n_u^p, n_{ur}^{pr})$ be the total weight of the shortest path in $\mathcal{G}^{(i-1)}$ from n_u^p to n_{ur}^{pr}, as kept in the table in node n_{ur}^{pr}. Specially, if the table in n_{ur}^{pr} does not contain an entry for n_u^p, it means $u_r \leq u$, and we define for this case $W^{(i-1)}(n_u^p, n_{ur}^{pr}) = \infty$. The total weight of the shortest path in $\mathcal{G}^{(i)}$ from n_u^p to n_i^l is then:

$$W^{(i)}(n_u^p, n_i^l) = \min[W^{(i-1)}(n_u^p, n_{ur}^{pr}) + w_{ur,i}^{pr,l}], \forall r \in [1, s] \tag{7.17}$$

where $w_{ur,i}^{pr,l}$ is the weight of edge $e_{ur,i}^{pr,l}$. The last node before n_i^l in the shortest path in $\mathcal{G}^{(i)}$ from n_u^p to n_i^l is n_{u*}^{p*}, where

$$\{u^*, p^*\} = \arg \min_{\{ur,pr\}} [W^{(i-1)}(n_u^p, n_{ur}^{pr}) + w_{ur,i}^{pr,l}], \forall r \in [1, s] \tag{7.18}$$

The two pieces of information for one entry in the table in node n_i^l are thus obtained: $W^{(i)}(n_u^p, n_i^l)$ and n_{u*}^{p*} are put into the entry for n_u^p. This process is applied to each node in $\mathcal{G}^{(i-1)}$, whereupon

Table 7.4: Solving the APSP in $\mathcal{G}^{(i)}$ using the solution of APSP in $\mathcal{G}^{(i-1)}$.

Assume: the APSP problem in $\mathcal{G}^{(i-1)}$ has been solved.

For each node $n_i^l \in \mathcal{N}_i$:

For each node $n_u^p \in \mathcal{G}^{(i-1)}$:

- add one entry labelled n_u^p to the table of APSP knowledge in n_i^l;
- put $W(n_u^p, n_i^l)$ and $n_{u^*}^{p^*}$ given by (7.17) and (7.18) into the entry.

the complete table in n_i^l is constructed. Since $\mathcal{G}^{(i)}$ is a semi-trellis, the shortest path in $\mathcal{G}^{(i-1)}$ between any pair of nodes in $\mathcal{G}^{(i-1)}$ is also the shortest path in $\mathcal{G}^{(i)}$ between the same pair of nodes. When the new node n_i^l and the associated edges $e_{u_1,i}^{p_1,l}, e_{u_2,i}^{p_2,l}, ..., e_{u_s,i}^{p_s,l}$ are added to $\mathcal{G}^{(i-1)}$, the tables in the nodes in $\mathcal{G}^{(i-1)}$ remain the same: neither do new entries need to be added to the tables, nor do the entries that are already in the tables need to be updated. This property of a semi-trellis means that, simply by applying the above process (see Algorithm 2 for a summation) as new nodes (and associated edges) are received, when the complete graph $\mathcal{G} = \mathcal{G}^{(K-V)}$ is constructed, the APSP problem in it is solved. The shortest path between any pair of nodes in $\mathcal{G}$ can be easily identified by back tracing.

Let h_i be the number of nodes in $\mathcal{N}_i$. The number of nodes in sub-graph $\mathcal{G}^{(i-1)}$ is then $\sum_{k=1+V}^{i-1} h_k$. To solve the APSP problem in $\mathcal{G}^{(i)}$, we need to construct a table of APSP knowledge for each node in $\mathcal{N}_i$. The number of operations of this process is in the order of $h_i \sum_{k=1+V}^{i-1} h_k$. The number of operations of the complete proposed APSP algorithm is then in the order of $\sum_{i=2+V}^{K-V}(h_i \sum_{k=1+V}^{i-1} h_k)$. Simple manipulation shows that the complexity of the proposed APSP algorithm is $O(N^2)$, where $N = \sum_{i=1+V}^{K-V} h_i$ is the total number of nodes in $\mathcal{G}$.

7.6.3 Path Level Analysis

Once the APSP problem is solved, the next step is to order the paths in $\mathcal{Q}$ according to their qualities, as defined in (7.16). For a typical sequence, the number of nodes N is of the order

of $10^2 \sim 10^3$. As a result, the number of paths Q is of the order of $10^4 \sim 10^6$. We use merge sort for its efficiency: the time complexity of merge sort is $O(Q \log Q)$ [11]. The paths in $\mathcal{Q}$ are stored in a linked list, and a merge sort implementation for linked list with improved memory efficiency [11] is used.

The next step of the complete algorithm is to apply the PR algorithm to get the best set of compatible path $\mathcal{B}$. To speed the process up, when checking the compatibility of a pair of paths, once a common support is found in both paths, the paths are declared incompatible.

7.7 Experiments Session I: Tracking a Single Ball Semi Automatically

Experimental Data and Experiment Setup In order to compare with the other two data association algorithms described in the previous chapters, in the first session of our experiments, the layered data association algorithm was used to track a single tennis ball in a semi-automatic fashion. That is, we used only the bottom two layers of the algorithm: candidate level association and tracklet level association. We manually specified the first and last ball-originated tracklets as the source and destination nodes, respectively, and used Dijkstra's algorithm to find the SPSP between them. The experimental data were the same as in the previous two chapters. The 70 sequences contain one play each. The sequences were split into a training set with 10 sequences and a test set with 60 sequences. For each SVM boundary, parameters of the tracker were tuned using the training set. The tracker with the optimal set of parameters was then tested using the test set.

Distribution of Tracking Errors and Proportion of LOT Frames From the distribution of tracking errors (see Fig. 6.6) and the proportion of LOT frames (see Table 6.6), we can see the performance of the tracker improves as the SVM boundary increase from -4 to -1. This is expected: as the number of false candidates drops, the number of false tracklets also drops. The SPSP is less likely to go through false nodes. As the SVM boundary continues to increase, however, the tracker's performance drops. This is because the algorithm requires three successive true candidates as seed triplet to grow a true tracklet. When the detection rate is low, some true

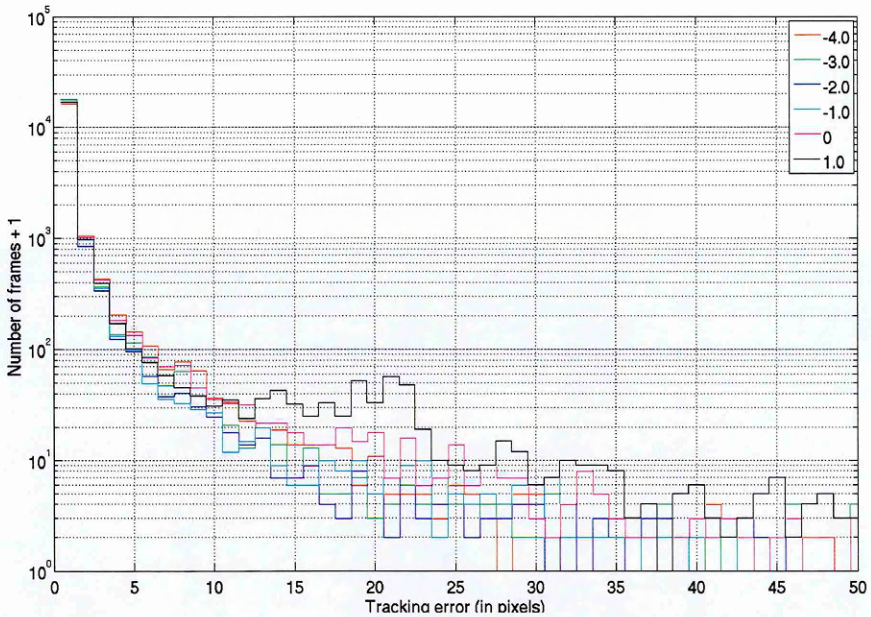


Figure 7.17: Distribution of tracking errors up to 50 pixels. The bin size is 1 pixels. Y-axis is the "number of frames + 1" in logarithmic scale.

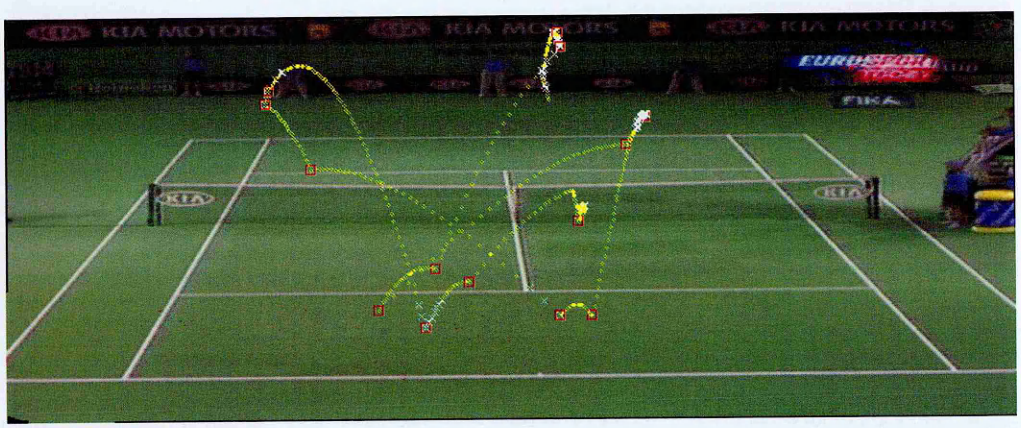
Table 7.5: Proportion of LOT frames.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
prop. of LOT Frames	**2.48%**	**1.78%**	**1.42%**	**1.43%**	**3.02%**	**5.22%**

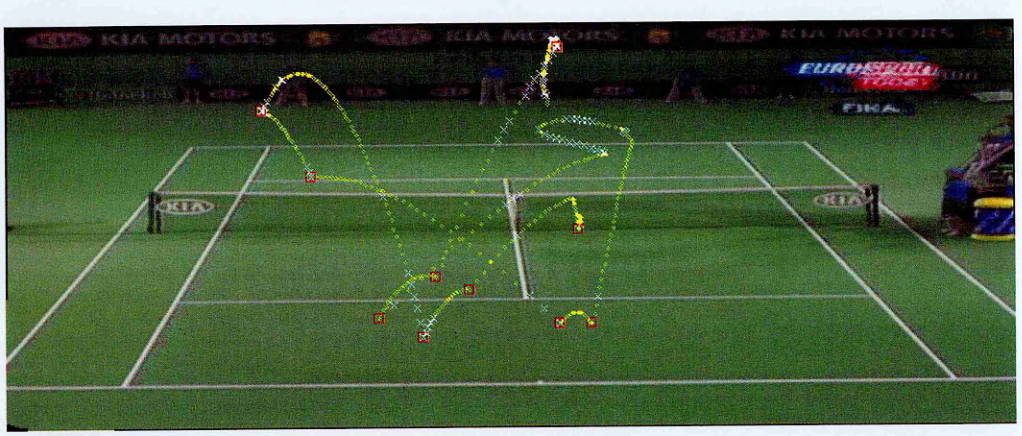
tracklets cannot be grown since the "seeds" cannot be formed. As a result, some ball-originated nodes are missing from the graph. The SPSP is "forced" to go through clutter-originated nodes (see Fig. 7.7 for an example).

Quality of Event Detection Table 7.7 shows the tracker's performance in terms of quality of event detection. Since event detection is directly affected by the tracking result, its quality also drops when r_d is low.

Processing Speed We can see from Table 7.7 that when SPSP is used at the tracklet level, the proposed algorithm is very efficient. On average, it runs approximately 10 times faster than the



(a)



(b)

Figure 7.18: The effect of the SVM boundary on the tracker's performance. (a) SVM boundary is -4. The tracker performs well despite the large number of false candidates. (b) SVM boundary is 0. Due to the low detection rate r_d, some ball-originated nodes are missing from the graph. The SPSP is "forced" to go through a clutter-originated node.

Table 7.6: Quality of event detection.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
ground truth	485					
detected	478	475	470	466	407	336
matched	386	383	384	381	291	202
recall	0.796	0.790	0.792	0.786	0.600	0.417
precision	0.808	0.806	0.817	0.818	0.715	0.601
F-measure	**0.802**	**0.798**	**0.804**	**0.801**	**0.653**	**0.492**

Table 7.7: Processing speed.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
frames per sec	**243.1**	**294.1**	**439.3**	**579.3**	**674.7**	**834.4**

other two algorithms. We can also see that as the SVM boundary increases, the speed of the algorithm also increases. This is because as the number of candidates drops, the number of seed triplets also drops. This not only reduces the time spent on growing tracklets, but also reduces the size of the constructed graph, which means it is faster to apply the Dijkstra's algorithm.

7.8 Experiments Session II: Tracking Multiple Balls Automatically

Experimental Data and Experiment Setup In the second session of our experiments, we used all three layers of the proposed algorithm: candidate level association, tracklet level association, and path level analysis. With the top layer of the algorithm, we can track multiple tennis balls fully automatically.

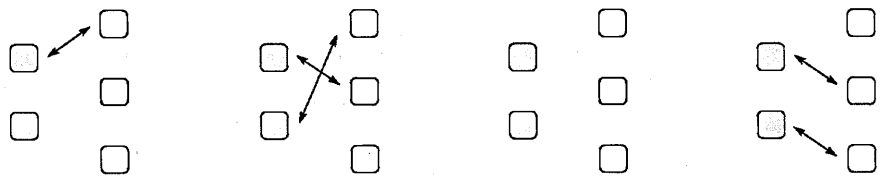


Figure 7.19: Four examples of valid assignment schemes. Shaded squares: trajectories in $\mathcal{T}_{gt}$. White squares: trajectories in $\mathcal{T}_{tr}$.

We used 70 sequences from the 2006 Australia Open tournament Men's final game. The number of plays in each sequence ranges from 2 to 4. In total the 70 sequences are approximately 20 minutes long, and contain 60,094 frames. The 70 sequences were split into a training set with 10 sequences containing 9,432 frames and a test set with 60 sequences containing 50,662 frames. Again, for each SVM boundary, parameters of the tracker were tuned using the training set to minimise the proportion of LOT frames. The tracker with the optimal set of parameters was then tested using the test set.

Automatically Matching the Tracked Trajectories and the Ground Truth Ones When multiple plays exist in one sequence, the correspondence between the reconstructed trajectories and the trajectories in ground truth is not straightforward. To automate the result analysis process, we need a method to automatically "match" the tracked trajectories and the ground truth trajectories.

Let $\mathcal{T}_{gt}$ be the set of trajectories in the ground truth, and $\mathcal{T}_{tr}$ be the set of trajectories given by the tracker. Let us define an assignment scheme as a correspondence between the two sets of trajectories, which satisfies the following constraint: each trajectory in $\mathcal{T}_{gt}$ corresponds to **at most** one trajectory in $\mathcal{T}_{tr}$, and vice versa. According to this constraint, it is possible that a trajectory in one set does not correspond to any trajectory in the other. Examples of valid assignment schemes are given in Fig. 7.8. Since in practice $\mathcal{T}_{gt}$ and $\mathcal{T}_{tr}$ contain small numbers of trajectories, it is possible to enumerate all assignment schemes.

Now we define the quality of an assignment scheme. First, observe that an assignment scheme consists of pairs of correspondences, where each pair is a correspondence between one trajectory in $\mathcal{T}_{gt}$ and one in $\mathcal{T}_{tr}$. For a given pair, let I^* be the time interval where the two trajectories overlap. For each frame in I^*, ball positions in both trajectories are compared. If the distance

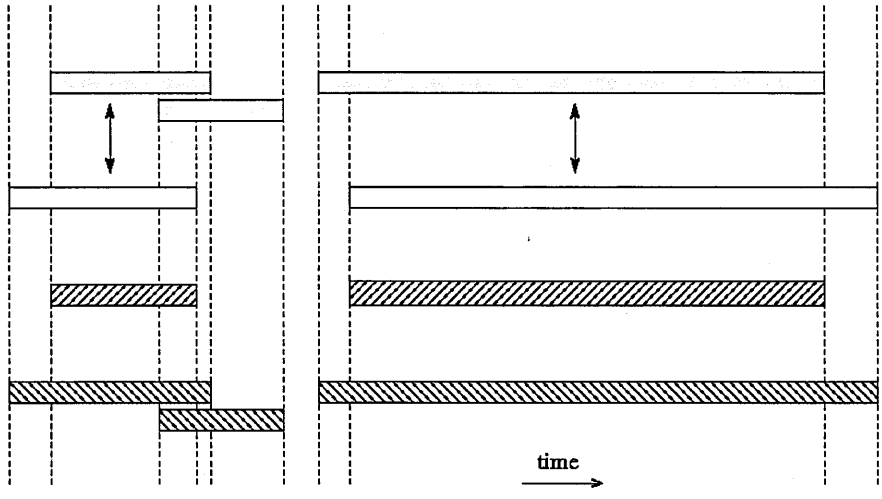


Figure 7.20: An illustrative example of the quality of track initiation and track termination. Each bar in the figure represents a time interval. The length of each bar is proportional to the length of the time interval. Shaded bars: time intervals of the trajectories in $\mathcal{T}_{gt}$. White bars: time intervals of the trajectories in $\mathcal{T}_{tr}$. Arrows: the best assignment scheme between $\mathcal{T}_{gt}$ and $\mathcal{T}_{tr}$. The total length of the bars with forward stripes is $I^{\cap}$, and the total length of the bars with backward stripes is $I^{\cup}$. The quality of track initiation and track termination, β, is defined as the ratio of $I^{\cap}$ to $I^{\cup}$.

between the positions is smaller than a threshold, the two trajectories are said to be "matched" in this frame. The match score of a pair of trajectories is defined as the total number of frames where the two trajectories are matched. The quality of an assignment scheme is then defined as the total match score of all its pairs of trajectories. Among all possible assignment schemes, the one with highest quality is chosen. A correspondence between the trajectories in $\mathcal{T}_{gt}$ and those in $\mathcal{T}_{tr}$ is thus obtained automatically.

Quality of Track Initiation and Track Termination Having obtained the correspondence between $\mathcal{T}_{gt}$ and $\mathcal{T}_{tr}$, we can measure the performance of the tracker from several aspects. In addition to the measures used for the other two data association algorithms, such as distribution of tracking errors, proportion of LOT frames, quality of event detection, and processing speed, since the proposed algorithm can deal with track initialisation/termination automatically, we would also like to measure the quality of track initiation and track termination.

The best assignment scheme consists of several pairs of correspondences. Let $T_{gt}^i \in \mathcal{T}_{gt}$ and $T_{tr}^i \in \mathcal{T}_{tr}$ be the trajectories in the i^{th} pair, I_{gt}^i and I_{tr}^i be the time interval of T_{tr}^i and T_{gt}^i,

Table 7.8: Quality of track initiation and track termination.

r_d	0.917	0.916	0.908	0.874	0.822	0.531
$\bar{N}$	12.2	9.0	5.1	0.9	0.1	0
$\bar{\beta}$	**85.94%**	**85.79%**	**86.94%**	**86.13%**	**81.22%**	**71.57%**

Table 7.9: Proportion of LOT frames.

r_d	0.917	0.916	0.908	0.874	0.822	0.531
$\bar{N}$	12.2	9.0	5.1	0.9	0.1	0
prop. of LOT Frames	**3.18%**	**3.11%**	**2.32%**	**2.06%**	**2.17%**	**2.91%**

respectively. In a general case, there may be one or more trajectories in $\mathcal{T}_{gt}$ and/or in $\mathcal{T}_{tr}$ that are not in any pair of correspondence. Let $T_{gt}^{'j}$ be the j^{th} trajectory in $\mathcal{T}_{gt}$ that is not in any pair, $I_{gt}^{'j}$ be its time interval; $T_{tr}^{'k}$ be the k^{th} trajectory in $\mathcal{T}_{tr}$ that is not in any pair, $I_{tr}^{'k}$ be its time interval. Now define

$$I^{\cap} = \sum_i |I_{tr}^i \cap I_{gt}^i| \tag{7.19}$$

and

$$I^{\cup} = \sum_i |I_{tr}^i \cup I_{gt}^i| + \sum_j |I_{gt}^{'j}| + \sum_k |I_{tr}^{'k}| \tag{7.20}$$

The quality of track initiation and track termination can then be defined as

$$\beta = \frac{I^{\cap}}{I^{\cup}} \tag{7.21}$$

where β ranges between 0 an 1. When $\beta = 1$, track initiation and termination are said to be perfect. An illustrative example of the definition of β is shown in Fig. 7.8.

The average performance of track initiation and termination on several sequences is defined as

$$\bar{\beta} = \frac{\sum_m I_m^{\cap}}{\sum_m I_m^{\cup}} \tag{7.22}$$

where $I_m^{\cap}$ and $I_m^{\cup}$ are the $I^{\cap}$ and $I^{\cup}$ on the m^{th} sequence, respectively. $\bar{\beta}$ on the 60 test sequences is shown in Table 7.8.

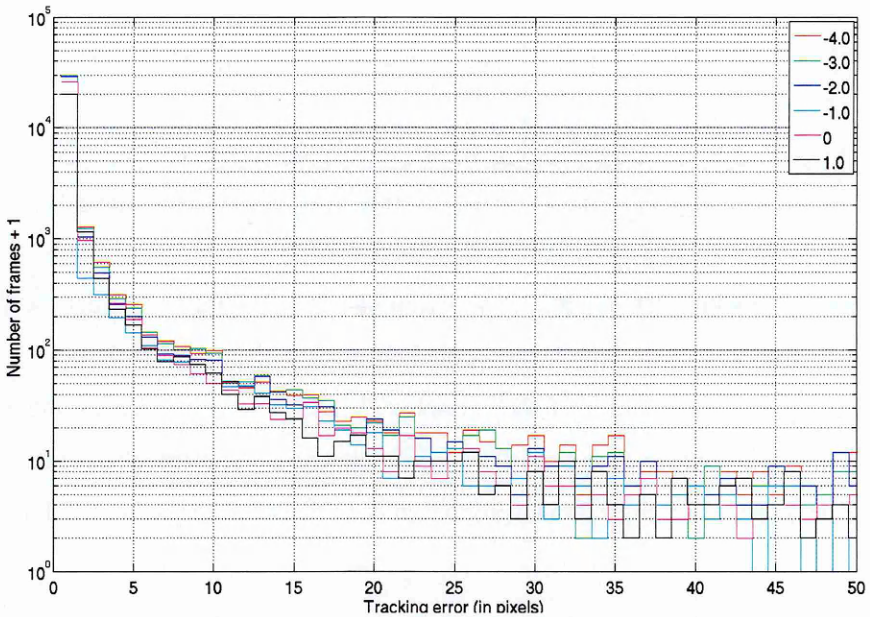


Figure 7.21: Distribution of tracking errors up to 50 pixels. The bin size is 1 pixels. Y-axis is the "number of frames + 1" in logarithmic scale.

Table 7.10: Quality of event detection.

r_d	0.917	0.916	0.908	0.874	0.822	0.531
$\bar{N}$	12.2	9.0	5.1	0.9	0.1	0
ground truth	717					
detected	721	734	700	703	674	564
matched	546	548	532	511	479	395
recall	0.762	0.764	0.742	0.713	0.668	0.551
precision	0.757	0.747	0.760	0.727	0.711	0.700
F-measure	**0.759**	**0.755**	**0.751**	**0.720**	**0.689**	**0.617**

Table 7.11: Processing speed.

r_d	0.917	0.916	0.908	0.874	0.822	0.531
$\bar{N}$	12.2	9.0	5.1	0.9	0.1	0
frames per sec	**46.3**	**59.4**	**72.8**	**93.6**	**116.2**	**142.4**

Distribution of Tracking Errors and Proportion of LOT Frames Consider the i^{th} pair of trajectories in the best assignment scheme. A tracking error is defined for each frame in $I^i_{tr} \cap I^i_{gt}$. For a given sequence, the number of such errors is $I^{\cap}$. For a set of sequences, the number of such errors is $\sum_m I^{\cap}_m$, where $I^{\cap}_m$ is the $I^{\cap}$ of the m^{th} sequence. The distribution of these $\sum_m I^{\cap}_m$ tracking errors is plotted in Fig 7.8. The proportion of LOT frames in shown in Table 7.8.

Quality of Event Detection A detected event in a tracked trajectory T^i_{tr} is regarded as "matched" if it is within 3 frames and 5 pixels of a ground truth event in the corresponding ground truth trajectory T^i_{gt}. The quality of event detection are shown in Table 7.8.

Processing Speed As the SVM boundary increases, the number of seed triplets also drops. This reduces the time spent on growing tracklets. At the same time, the size of the constructed graph is reduced, which means the proposed APSP algorithm, which has an $O(N^2)$ time complexity, can also run faster.

Parameter values used in both sessions of the experiments were: $V = 15$ frames, $R = 25.0$ pixels, $d_{th} = 5.0$ pixels, $m_{th} = 6$, $k_{th} = 25$ frames, $L_{th} = 30$, and $\alpha = 1.0$ pixel.

7.9 Conclusions

In this chapter, we proposed a multi-layered data association scheme with graph-theoretic formulation. There is a natural connection between the optimal path problem in a graph and the data association problem. The Viterbi-based data association algorithm presented in Chapter 6 is also a graph-theoretic algorithm. In the Viterbi-based algorithm, each object candidate is modelled as a node in a graph, and the graph constructed is a trellis. The algorithm used for searching

the optimal path is the Viterbi algorithm. The main difference of the method described in this chapter from that in Chapter 6 is that, before constructing a graph, we map the data association problem from a candidate space to a tracklet space, where it is easier to deal with. We then further map the association problem onto a graph, and solve it innovatively using an all-pairs shortest path formulation and path level analysis. This results in a fully automatic algorithm, which is capable of tracking multiple objects. By exploiting a special topological property of the graph, we have also developed a more efficient APSP algorithm than the general-purpose ones.

Chapter 8

Comparison of the Algorithms and Discussion

8.1 Algorithms for Comparison

For comparison with the proposed algorithms, the Probabilistic Data Association (PDA) [2] and the Robust Data Association (RDA) [36] were also implemented. Since PDA by itself can not deal with abrupt motion changes, in our experiments, it lost track in most sequences. This happened mostly when the near player hit the ball. This is because, in the image plane, the motion change of the ball is more drastic when the ball is hit by the near player than by the far player. The loss of track could not be recovered since no track reinitialisation mechanism was used. Since PDA failed completely, its performance is not shown here. We will focus on the comparison of the three proposed methods and RDA.

8.2 Performance of RDA

The experimental data were the same as in Section 5.7, Section 6.6, and Section 7.7. In RDA, the number of trials, N_t, is chosen so that the probability of finding a set that consists entirely of true positives is greater than a threshold P. In our experiments, P was set to 0.99. Once P was set, N_t could be computed according to (3.38). Table 8.2 shows the required number of trials N_t

Table 8.1: Number of trials required by RDA.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
N_t	**15762**	**8977**	**783**	**19**	**4**	**1**

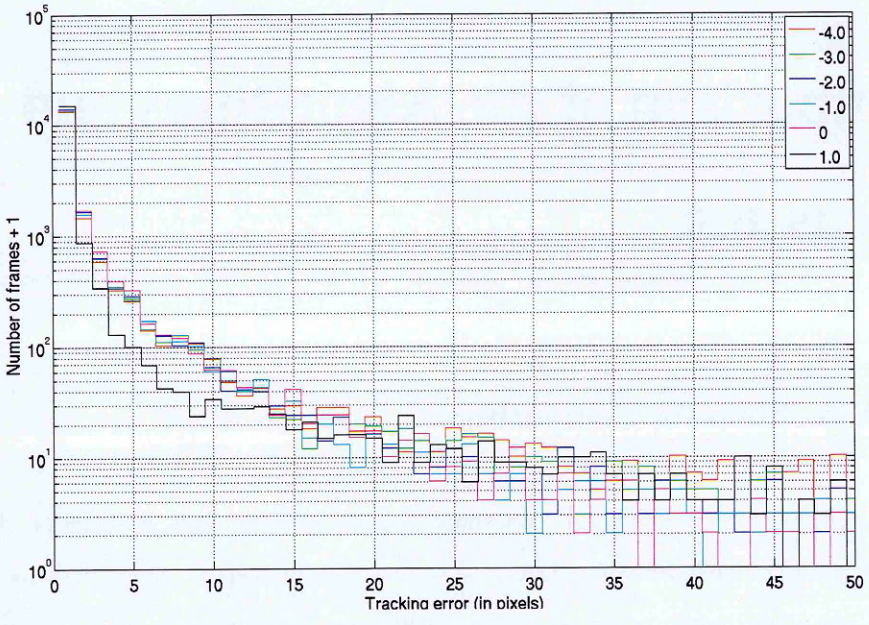
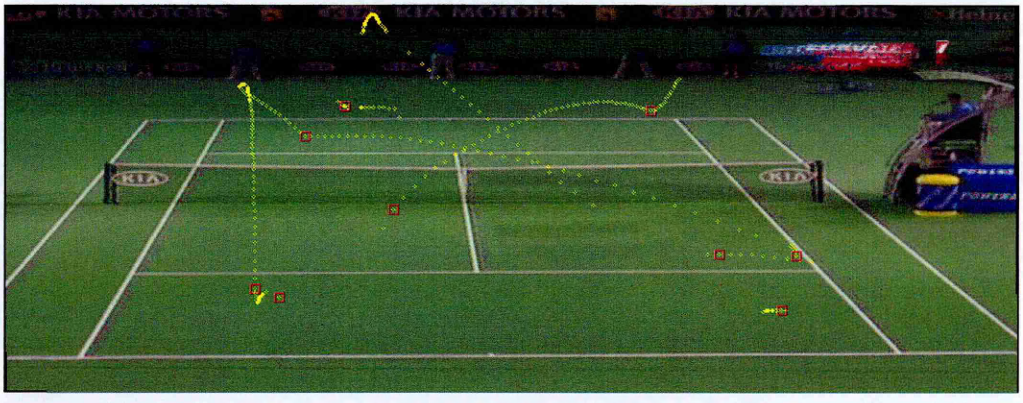


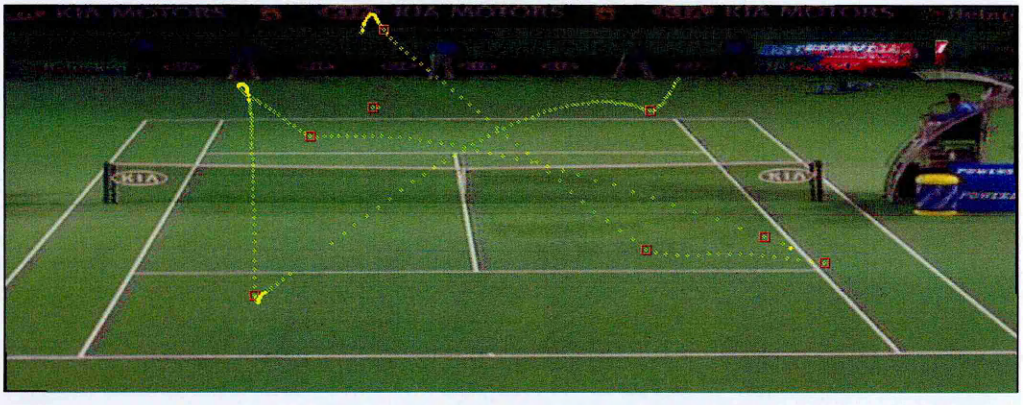
Figure 8.1: Distribution of tracking errors up to 50 pixels. The bin size is 1 pixels. Y-axis is the "number of frames + 1" in logarithmic scale.

under various clutter level and detection rate. In our experiments, the interval size of RDA was set to 15, as suggested in [36]. The mixture parameters of the likelihood function in RDA was estimated recursively until convergence, also as suggested in [36]. The dynamic model fitted to a set of three candidates was the same as used in the layered data association algorithm: a constant acceleration model as defined in (7.1) and (7.2). RDA by itself cannot deal with track initiation and track termination. In our experiments, we manually specified the frames in which it was applied.

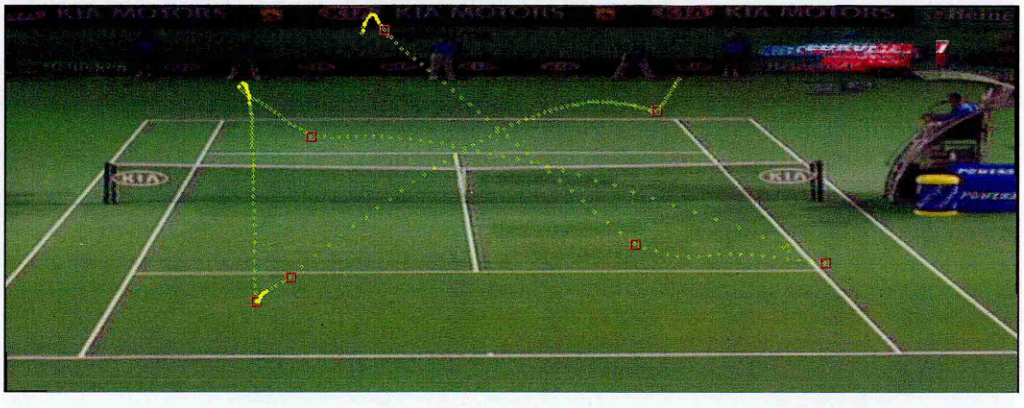
As usual, we show the distribution of tracking errors, the proportion of LOT frames, the quality of event detection, and the processing speed in Fig. 8.2, Table 8.2, Table 8.2, and Table 8.2, respectively. It can be seen from Fig. 8.2 and the tables that as the SVM boundary increases



(a)



(b)



(c)

Figure 8.2: The effect of the SVM boundary on the tracker's performance. (a) and (b): SVM boundary is -4 and -2, respectively. In some sliding window positions, a model fitted to false candidates that have originated from parts of the players can win the competition. (c) SVM boundary is 0. Models fitted to false candidates are now unlikely to win the competition.

Table 8.2: Proportion of LOT frames.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
prop. of LOT Frames	**11.28%**	**8.80%**	**6.04%**	**5.34%**	**4.80%**	**6.54%**

Table 8.3: Quality of event detection.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
ground truth	485					
detected	544	527	485	468	426	334
matched	206	221	240	219	246	189
recall	0.425	0.456	0.495	0.452	0.507	0.390
precision	0.379	0.419	0.495	0.467	0.578	0.566
F-measure	**0.400**	**0.437**	**0.495**	**0.459**	**0.540**	**0.462**

from -4 to 0, the performance of the RDA algorithm both in terms of tracking error and in terms of event detection improves. This might not seem to be what we would have expected. Although the detection rate and the number of false candidates both vary as the SVM boundary shifts, so does the number of trials. As discussed earlier, the number of trials N_t is chosen so that the probability of getting at least one sample set consisting entirely of true positives is 0.99. As a result, the tracker's performance "should" stay the same as the SVM boundary moves. However, we will show this is a conclusion based on a false assumption.

In RDA, or more generally in RANSAC, we make the hidden assumption that a model given by an uncontaminated sample set is always "better" than that given by a contaminated sample set. This is not true when applying RDA to tennis ball tracking. As previously mentioned, in a tennis video, the ball is not the only smoothly moving object. Candidates that have originated from

Table 8.4: Processing speed.

r_d	0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$	13.1	10.7	4.2	0.6	0.1	0
frames per sec	**1.4**	**2.5**	**27.4**	**279.0**	**400.4**	**453.7**

part of a player, *e.g.* a wrist band, can form smooth trajectories. In other words, false candidates can also be well fitted to a constant acceleration model. Consequently, a model given by an uncontaminated sample set can "lose" in the competition with a model given by a contaminated set.

We can clearly see this in Fig. 8.2. When SVM boundary is between -4 and -2, a large number of false candidates exist. In some sliding window positions, a model fitted to false candidates that have originated from parts of the players can win the competition. The estimated position given by this model is then used as the tracked ball position. Since RDA is completely non-recursive, that is, estimates are given by the best models in corresponding intervals independently of each other, an estimate given by a wrong model in general can be very far from the true position of the ball. This explains the large tracking errors in Fig. 8.2 when the SVM boundary is low.

As the SVM boundary increases, the number of false candidates drops significantly, including the ones that can form smooth motions. As a result, models given by false candidates become less and less likely to win the competition with those given by true candidates. This explains the performance improvement as the SVM boundary increases. When the SVM boundary is 1, the detection rate is very low. The drop in performance is due to inaccurate interpolation.

We can see in Table 8.2 that the processing speed of RDA increases by more than 300 fold as the SVM boundary increases from -4 to 1. This is expected, since the speed of RDA is determined by the number of trials N_t, which is in turn determined by r_d and $\bar{N}$, according to (3.38).

Table 8.5: Proportion of LOT frames.

r_d		0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$		13.1	10.7	4.2	0.6	0.1	0
prop. of LOT Frames	Particle	**2.21%**	**2.26%**	**3.18%**	**12.48%**	**14.11%**	**34.01%**
	Viterbi	**2.27%**	**2.08%**	**1.44%**	**1.51%**	**1.90%**	**4.11%**
	Layered	**2.48%**	**1.78%**	**1.42%**	**1.43%**	**3.02%**	**5.22%**
	Robust	**11.28%**	**8.80%**	**6.04%**	**5.34%**	**4.80%**	**6.54%**

8.3 Comparison of the Algorithms and Discussion

Previously in this thesis, for each of the four data association algorithms, we have presented and discussed its performance under various clutter levels and detection rates. In other words, we have compared the performance of the algorithms "horizontally". In this section, we present the same experimental results, but from another perspective. For each clutter level and detection rate, we discuss the performance of the four algorithms. In other words, this time we compare the performance of the algorithms "vertically". We also discuss the reasons behind the different performances of the algorithms. For convenience, in the rest of the discussion, we call the particle filter based data association algorithm MDA ("M" for Monte-Carlo, in order to avoid confusion with the Probabilistic Data Association). Similarly, we call the Viterbi based algorithm VDA, and the layered data association algorithm LDA, respectively.

Distribution of Tracking Errors and Proportion of LOT Frames Fig 8.3 and Table 8.3 compare the distribution of tracking errors and the proportion of LOT frames of the four algorithms under each SVM boundary, respectively. Very briefly speaking, the VDA and the LDA outperform the other two algorithms. When looking more carefully at Fig 8.3 and Table 8.3, we can see the four algorithms have different characteristics.

When clutter level and detection rate are both high, the performance of the MDA is comparable to that of VDA and LDA. However, as the clutter level and detection rate drop, its performance

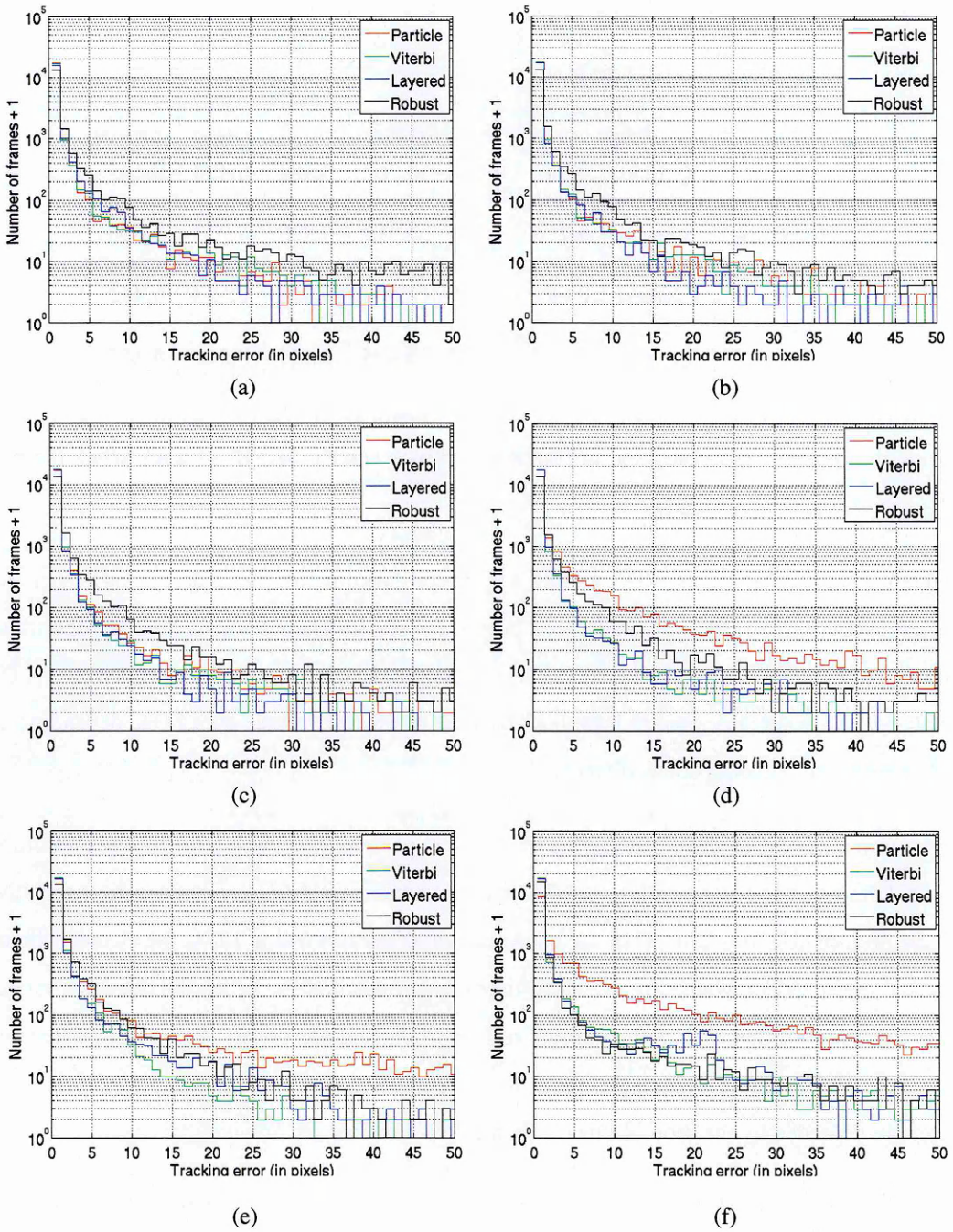


Figure 8.3: Distributions of tracking errors up to 50 pixels of the four data association algorithms. From (a) to (f): SVM decision boundary is set to -4, -3, -2, -1, 0, 1, respectively.

Table 8.6: Quality of event detection.

r_d		0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$		13.1	10.7	4.2	0.6	0.1	0
F-measure	Particle	**0.769**	**0.786**	**0.762**	**0.581**	**0.450**	**0.245**
	Viterbi	**0.778**	**0.789**	**0.813**	**0.820**	**0.753**	**0.646**
	Layered	**0.802**	**0.798**	**0.804**	**0.801**	**0.653**	**0.492**
	Robust	**0.400**	**0.437**	**0.495**	**0.459**	**0.540**	**0.462**

drops significantly. As discussed in Chapter 5, this is due to the loss of track when the ball changes its motion drastically.

Among the four algorithms, VDA gives the most consistent performance across various clutter levels and detection rates. It has the lowest proportion of LOT frames among the four algorithms in two configurations, and the second lowest in another three configurations. In terms of LOT analysis, its overall performance is perhaps the best, or the second best after LDA, depending on the definition of "overall" performance.

LDA is another strong candidate for the best overall performance, with three lowest proportion of LOT frames, and two second lowest. The main disadvantage of LDA is that it is more sensitive to the drop of detection rate. This can be explained by the fact that in LDA, the elements being tracked are tracklets, which are grown from triplets of candidates. When the detection rate of true candidates drops, the "detection rate of true triplets" drops faster. As a result, compared to VDA, where candidates are the elements being tracked, the performance of LDA is more severely degraded by the drop of true candidate detection rate.

In our experiments, RDA performed poorly. On the surface, this is because RDA makes the wrong assumption that, in each interval, there is only one smooth motion that can be explained by the dynamic model, and that smooth motion has originated from the object being tracked. Deep down, however, RDA's poor performance can be explained by its low utilisation efficiency of available information.

Table 8.7: Features of the four algorithms.

	fully automatic	track multiple balls	online processing	best LOT perf.‡
Particle	×	×	×†	2.21
Viterbi	×	×	×	1.44
Layered	√	√	×	1.42
Robust	×	×	√	4.80

† However, the particle filter based method can operate online if the smoothing process is not used.
‡ Best performance in terms of proportion of LOT frames in all 6 configurations.

In offline data association, all the available information is the object candidate positions (measurements) in all the frames in the sequence. RDA is sliding-window-based and non-recursive, meaning that only information inside the interval is used for resolving data association ambiguity, information outside the interval is completely discarded. As a result, associations are given by the best models in corresponding intervals independently of each other. Once a clutter-originated motion is wrongly picked up as the best model in an interval, there is no mechanism to recover from such an error. By contrast, all the three proposed algorithms somehow utilise information in all frames when making association decision in each frame: in MDA, this is achieved with the smoothing pass; in VDA, this is achieved by delaying the association decision; and in LDA, this is provided by the top two layers, the APSP at the tracklet level and the path reduction algorithm at the path level.

On the other hand, perhaps it is unfair to compare RDA with the three proposed algorithms, since RDA is the only online algorithm among the four. In RDA, the estimated ball position in a frame is reported with a small delay after the measurements in this frame are obtained. By contrast, the MDA is a multi-scan offline algorithm, with the particle filtering as the first scan and smoothing (and data association) as the second scan; the VDA is a single scan offline algorithm where the association in a frame is not reported until the last frame in the sequence is reached; and the LDA is also an offline algorithm, where, again, the association result in a frame is not reported until the last frame in the sequence is reached. Features of the four algorithms are summarised in Table 8.3.

Table 8.8: Processing speed.

r_d		0.931	0.931	0.924	0.903	0.826	0.574
$\bar{N}$		13.1	10.7	4.2	0.6	0.1	0
frame per sec	Particle	**26.3**	**28.0**	**32.1**	**33.8**	**35.5**	**36.3**
	Viterbi	**42.3**	**46.3**	**50.1**	**51.2**	**52.0**	**53.4**
	Layered	**243.1**	**294.1**	**439.3**	**579.3**	**674.7**	**834.4**
	Robust	**1.4**	**2.5**	**27.4**	**279.0**	**400.4**	**453.7**

Quality of Event Detection Table 8.3 compare the quality of event detection of the four algorithms. The quality of event detection is affected directly by the quality of the tracking. As a result, the trend shown in Table 8.3 coincides well with that shown in Fig 8.3 and Table 8.3, and can be explained by the same arguments.

Processing Speed In Table 8.3, the speed of four algorithms is compared. This time, LDA is the winner without any doubt. The fact that LDA always starts model fitting from a seed triplet — three candidates that have high probability of containing only true positives — allows it to eliminate false candidates very quickly. Once triplets are grown into tracklets, the data association problem becomes a shortest path problem in a much smaller graph, compared to the graph that would have been constructed in VDA. It should be noted that as the SVM boundary increases, RDA has the fastest growing processing speed. This is because the time complexity of RDA is determined directly by the number of trials evaluated, which drops rapidly as the proportion of true positives increases (see Table 8.2).

8.4 Conclusions

In this chapter, we compared the performances of the proposed data association algorithms, both mutually and with that of the RDA. Experiments show that the four algorithms have different failure modes. Generally speaking, the particle filter based algorithm performs well when the

clutter level and the detection rate are both high, while performs poorly when they are both low. On the other hand, the RDA has the very opposite characteristic. Overall, the Viterbi based algorithm and the layered algorithm outperform the particle filter algorithm and the RDA. When making comparison between them, the layered algorithm is slightly more sensitive to the drop of detection rate than the Viterbi based algorithm, but is computationally much more efficient than all the other three algorithms. The reasons behind the different characteristics of the algorithms were also discussed.

Chapter 9

Summary and Future Work

9.1 Summary of the Thesis

This thesis investigated computer vision techniques for tracking tennis balls in broadcast tennis videos. We argue that the Track-After-Detection (TAD) type of approach is suitable for the tennis ball tracking problem, and ball candidate detection and data association are the two key sub-problems. In this thesis, we have developed a novel ball candidate detection algorithm, and three novel data association algorithms.

We started in Chapter 1 with an introduction to the thesis, including motivation of this work, difficulties of the task, and the contributions. The objective of the work presented in this thesis was to provide a tennis ball tracking module for an automatic tennis video annotation system. In Chapter 2, this annotation system was briefly introduced. Next, in Chapter 3, techniques related to this thesis, including those for data association and estimation, the sequential Monte-Carlo method, the RANSAC algorithm, and some existing tennis ball tracking algorithms, were reviewed.

From Chapter 4 to Chapter 7, we presented the four proposed algorithms, one algorithm in each chapter. In Chapter 4, the ball candidate detection algorithm was described. This algorithm takes as input video frames, and outputs ball candidate positions for the following data association step. It extracts foreground moving blobs using temporal differencing, and then classify the blobs into ball candidates and non-candidates. A novel gradient-based feature for measuring the

departure of the shape of a blob from an ellipse is proposed.

The first data association algorithm, which is based on particle filtering, was described in Chapter 5. We proposed a new sampling scheme with an improved efficiency. In order to meet the requirement of our video annotation application, we also used a combination of smoothing and an explicit data association process to increase the accuracy of the tracking results.

The particle filter based algorithm solves the data association problem in an "indirect" way, in that state estimation and object-candidate association are dealt with sequentially. In Chapter 6, we proposed an algorithm that solved the data association problem directly. The algorithm seeks the sequence of object-measurement associations with maximum *a posteriori* probability, using a modified version of the Viterbi algorithm.

Inspired by the Robust Data Association (RDA)'s model-fitting approach to the data association problem, and by the conclusion that all available information should be exploited as fully as possible for resolving data association ambiguity, in Chapter 7, we presented a layered data association algorithm with graph-theoretic formulation. This algorithm slices the data association problem into three layers, and tackle the three sub-problems using: candidate level association, tracklet level association, and path level analysis, respectively. The complete algorithm can track multiple objects. It can also deal with track initiation and track termination automatically.

In Chapter 8, the performances of the three proposed data association algorithms were compared, both mutually and against those of some existing algorithms. The advantages and disadvantages of the algorithms were discussed. Experiments show that the particle filter based algorithm gives similar performance to RDA, which is the state-of-the-art tennis ball tracking method for monocular sequences; while the other two proposed algorithm both outperform RDA. When taking into account the various performance benchmarks, the layered algorithm with graph-theoretic formulation in Chapter 7 emerges as the winner. It performs well in terms of both tracking accuracy and quality of event detection, and is computationally very efficient. It also has the added advantage of being able to track multiple objects and to deal with track initiation/termination automatically.

9.2 Future Work

As mentioned at the beginning of this thesis, we have decided to treat the tracking problem as a pure data association problem. That is, we use **only** the positions of the ball candidates for the tracking. In ball candidate detection, the SVM classifier provides for each foreground blob a likelihood of it being ball-originated. Only blobs with likelihoods higher than a threshold are used as ball candidates, and their likelihoods are ignored in the following data association process. One possible direction for future research is then to somehow utilise the likelihoods of the candidates, and see if data association can benefit from this piece of information.

In the layered data association algorithm described in Chapter 7, the generated tracklets are first thresholded based on the number of supports it has, or its "strength". Only tracklets that are "strong enough" are modelled as nodes in a graph. We then weight each edge, that is, we assign a real-valued "distance" to each edge, which represents the compatibility of the two nodes joined by the edge. Tracklet level association is then formulated as a shortest path problem. This formulation, however, involves a thresholding process, and ignores the strengths of the tracklets that have passed the thresholding. A more elegant formulation would be to weight the nodes in the graph as well based on their strengths. The association problem can then be transformed to an optimal path problem under multiple constraints. For example, we can then look for a path with both short total distance and high total strength.

The proposed algorithms have only been tested on tennis sequences. In the future, we would like to extend them to track balls in other ball games, such as a football or a cricket ball. We would also like to extend them to serve as generic trackers for tracking small and fast moving objects that undergo "switching" dynamics.

Bibliography

[1] Ravl: Recognition and vision library. http://www.ee.surrey.ac.uk/CVSSP/Ravl.

[2] Y. Bar-Shalom and T. E. Fortmanm. *Tracking and Data Association.* Academic Press INC., 1988.

[3] Y. Bar-shalom, X. Li, , and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, 2001.

[4] S. S. Blackman, M. T. Busch, and R. F. Popoli. Imm-mht tracking and data association for benchmark tracking problem. In *American Control Conference*, volume 4, pages 2606–2610, 1995.

[5] D. Capel. An effective bail-out test for ransac consensus scoring. In *British Machine Vision Conference*, volume 2, pages 629–638, 2005.

[6] W. Christmas, A. Kostin, F. Yan, I. Kolonias, and J. Kittler. A system for the automatic annotation of tennis matches. In *Fourth International Workshop on Content-Based Multimedia Indexing*, 2005.

[7] O. Chum and J. Matas. Randomized ransac with t(d, d) test. In *British Machine Vision Conference*, 2002.

[8] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 220–226, 2005.

[9] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *DAGM-Symposium*, pages 236–243, 2003.

[10] O. Chum, J. Matas, and S. Obdrzalek. Enhancing ransac by generalized model optimization. In *Asian Conference on Computer Vision*, volume 2, pages 812–817, 2004.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[12] I. J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[13] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[14] S. Davey. Extensions to the probabilistic multi-hypothesis tracker for improved data association. PhD Thesis, 2003.

[15] S. Davey, B. Colegrove, and D. Gray. A comparison of track initiation with pdaf and pmht. In *IEE Radar Conference*, pages 320–324, 2002.

[16] M. Delakis, G. Gravier, and P. Gros. Multimodal segmental-based modeling of tennis video broadcasts. In *IEEE International Conference on Multimedia and Expo*, pages 546–549, 2005.

[17] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[18] H. Denman, N. Rea, and A. Kokaram. Content based analysis for video from snooker broadcasts. In *International Conference on Image and Video Retrieval*, pages 198–205, 2002.

[19] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[20] M. Efe, Y. Ruan, and P. Willett. Probabilistic multi-hypothesis tracker: Addressing some basic issues. *IEE Proceedings on Radar, Sonar and Navigation*, 151(4):189–196, 2004.

[21] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association for Computing Machinery*, 24(6):381–395, 1981.

[22] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962.

[23] G. Fomey. The viterbi algorithm. *Proceedings of the IEEE*, pages 268–278, 1973.

[24] T. E. Fortmann, Y. Bar-Shalom, M. Scheffe, and S. Gelfand. Detection thresholds for tracking in clutter - a connection between estimation and signal processing. *IEEE Transactions on Automatic Control*, 1985.

[25] R. Halir and J. Flusser. Numerically stable direct least squares fitting of ellipses. In *International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media*, pages 125–132, 1998.

[26] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004.

[27] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal on Computer Vision*, 1(29):5–28, 1998.

[28] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *European Conference on Computer Vision*, volume 1, pages 893–908, 1998.

[29] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977.

[30] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[31] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[32] J. Kittler, W. Christmas, A. Kostin, F. Yan, I. Kolonias, and D. Windridge. A memory architecture and contextual reasoning framework for cognitive vision. In *14th Scandinavian Conference on Image Analysis*, volume 3540, page 343, 2005.

[33] A. Kokaram, F. Pitie, R. Dahyot, N. Rea, and S. Yeterian. Content controlled image representation for sports streaming. In *IEEE Workshop on Content Based Multimedia Indexing*, 2005.

[34] I. Kolonias, W. Christmas, and J. Kittler. A layered active memory architecture for cognitive vision systems and its application to sport video evolution tracking. In *International Conference on Computer Vision Systems*, 2007.

[35] I. Kolonias, F. Yan, W. Christmas, A. Kostin, and J. Kittler. A contextual reasoning framework for scene interpretation and tracking in tennis video sequences. To appear in Computer Vison and Image Understanding Special Issue on Computer Vision Based Analysis in Sport Environments.

[36] V. Lepetit, A. Shahrokni, and P. Fua. Robust data association for online applications. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 281–288, 2003.

[37] J. Matas and O. Chum. Randomized ransac with sequential probability ratio test. In *IEEE International Conference on Computer Vision*, 2005.

[38] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: A survey. *IEEE Transactions on Aerospace and Electronic Systems*, 34(1):103–122, 1998.

[39] H. Miyamori and S. Iisaku. Video annotation for content-based retrieval using human behavior analysis and domain knowledge. In *International Conference on Automatic Face and Gesture Recognition*, pages 320–325, 2000.

[40] C. L. Morefield. Application of 0-1 integer programming to mutitarget tracking problems. *IEEE Transactions on Automatic Control*, 22(3):302–312, 1971.

[41] V. Mottl, A. Kostin, and I. Muchnik. Generalized edge-preserving smoothing for signal analysis. In *IEEE Workshop on Nonlinear Signal and Image Analysis*, 1997.

[42] D. Musicki, R. Evans, and S. Stankovic. Integrated probabilistic data association. *IEEE Transactions on Automatic Control*, 39(6):1237–1241, 1994.

[43] D. Musicki, B. F. La Scala, and R. J. Evans. Integrated track splitting filter for manoeuvring targets. In *IEEE International Conference on Information Fusion*, 2004.

[44] D. R. Myatt, P. H. S. Torr, S. J. Nasuto, J. M. Bishop, and R. Craddock. Napsac: High noise, high dimensional robust estimation - it's in the bag. In *British Machine Vision Conference*, pages 458–467, 2002.

[45] K. Nummiaro, E. K. Merier, and L. V. Gool. An adaptive color-based particle filter. *Journal of Image and Vision Computing*, 21(1):99–110, 2003.

[46] T. Ogata, W. Christmas, J. Kittler, and S. Ishikawa. Tennis stroke detection and classification based on boosted activity detectors and particle filtering. In *International Conference on Soft Computing and Intelligent Systems and International Symposium on Advanced Intelligent Systems (SCIS&ISIS)*, pages 2035–2040, 2006.

[47] N. Owens, C. Harris, and C. Stennett. Hawk-eye tennis system. In *International Conference on Visual Information Engineering*, 2003.

[48] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, pages 661–675, 2002.

[49] M. Petkovic, V. Mihajlovic, W. Jonker, and S. Djordjevic-Kajan. Multi-modal extraction of highlights from tv formula 1 programs. In *IEEE International Conference on Multimedia and Expo*, volume 1, pages 817–820, 2002.

[50] G. S. Pingali, Y. Jean, and I. Carlbom. Real time tracking for enhanced tennis broadcasts. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 260–265, 1998.

[51] G. S. Pingali, A. Opalach, and Y. Jean. Ball tracking and virtual replays for innovative tennis broadcasts. In *IEEE International Conference on Pattern Recognition*, pages 4152–4156, 2000.

[52] M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94:590–623, 1999.

[53] G. W. Pulford. Taxonomy of multiple target tracking methods. *IEE Proceedings on Radar, Sonar and Navigation*, 152(5):291–304, 2005.

[54] G. W. Pulford and A. Logothetis. An expectation-maximisation tracker for multiple observations of a single target in clutter. In *IEEE Conference on Decision and Control*, volume 5, pages 4997–5003, 1997.

[55] T. Quach and M. Farooq. Maximum likelihood track formation with the viterbi algorithm. In *IEEE Conference on Dedsion and Control*, pages 271–276, 1994.

[56] C. Rago, P. Willett, and R. Streit. A comparison of the jpdaf and pmht tracking algorithms. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 3571–3574, 1995.

[57] C. Rago, P. Willett, and R. Streit. Direct data fusion using the pmht. In *American Control Conference*, volume 3, pages 1698–1702, 1995.

[58] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[59] B. Ristic, A. Arulampalam, and N. Gordon. *Beyond the Kalman Filter - Particle Filters for Tracking Applications*. Artech House, 2004.

[60] M. Roh, W. Christmas, J. Kittler, and S. Lee. Robust player gesture spotting and recognition in low-resolution sports video. In *9th European Conference on Computer Vision*, volume 4, pages 347–358, 2006.

[61] Y. Ruan, P. Willett, and R. Streit. A comparison of the pmht and pdaf tracking algorithms based on their model crlbs. In *SPIE Aerosense Conference on Acquisition, Tracking and Pointing*, 1999.

[62] D. Salmond. Mixture reduction algorithms for target tracking in clutter. In *SPIE Conference on Signal and Data Processing of Small Targets*, volume 1305, pages 434–445, 1990.

[63] P. Smith and G. Buechler. A branching algorithm for discriminating and tracking multiple objects. *IEEE Transactions on Automatic Control*, pages 101–104, 1975.

[64] R. Streit and T. Luginbuhl. Probabilistic multi-hypothesis tracking. Technical Report, 1995.

[65] B. J. Tordoff and D. W. Murray. Guided-mlesac: Faster image transform estimation by using matching priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1523–1535, 2005.

[66] P. H. S. Torr and C. Davidson. Impsac: Synthesis of importance sampling and random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):354–364, 2003.

[67] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 2000.

[68] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1999.

[69] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[70] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report, 2004.

[71] P. Willett, Y. Ruan, and R. Streit. The pmht for maneuvering targets. In *SPIE Conference on Signal and Data Processing of Small Targets*, pages 416–427, 1998.

[72] P. Willett, Y. Ruan, and R. Streit. Pmht: Problems and some solutions. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):738–754, 2002.

[73] J. L. Williams. Gaussian mixture reduction for tracking multiple maneuvering targets in clutter. Master's Thesis, 2003.

[74] J. K. Wolf, A. M. Viterbi, and S. G. Dixon. Finding the best set of k paths through a trellis with application to multitarget tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 25:287–296, 1989.

[75] F. Yan, W. Christmas, and J. Kittler. A tennis ball tracking algorithm for automatic annotation of tennis match. In *British Machine Vision Conference*, volume 2, pages 619–628, 2005.

[76] F. Yan, W. Christmas, and J. Kittler. A maximum a posteriori probability viterbi data association algorithm for ball tracking in sports video. In *IEEE International Conference on Pattern Recognition*, 2006.

[77] F. Yan, W. Christmas, and J. Kittler. All pairs shortest path formulation for multiple object tracking with application to tennis video analysis. In *British Machine Vision Conference*, 2007.

[78] F. Yan, A. Kostin, W. Christmas, and J. Kittler. A novel data association algorithm for object tracking in clutter with application to tennis video analysis. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2006.

[79] X. Yu, H. W. Leong, J. Lim, Q. Tian, and Z. Jiang. Team possession analysis for broadcast soccer video based on ball trajectory. In *IEEE International Conference on Information, Communications and Signal Processing*, volume 3, pages 1811–1815, 2003.

[80] X. Yu, C. Sim, J. R. Wang, and L. Cheong. A trajectory-based ball detection and tracking algorithm in broadcast tennis video. In *International Conference on Image Processing*, volume 2, pages 1049–1052, 2004.