

# Chapter 2

## Ball Tracking for Tennis Video Annotation

Fei Yan, William Christmas and Josef Kittler

**Abstract** Tennis game annotation using broadcast video is a task with a wide range of applications. In particular, ball trajectories carry rich semantic information for the annotation. However, tracking a ball in broadcast tennis video is extremely challenging. In this chapter, we explicitly address the challenges, and propose a layered data association algorithm for tracking multiple tennis balls fully automatically. The effectiveness of the proposed algorithm is demonstrated on two data sets with more than 100 sequences from real-world tennis videos, where other data association methods perform poorly or fail completely.

### 2.1 Introduction

Effective and efficient sports video annotation systems have wide applications in, e.g. content-based video retrieval, enhanced broadcast, summarisation, object-based video encoding, automatic analysis of player tactics, to name a few. Owing to advances in computer vision and machine learning, building such tools has become possible [10, 13, 15].

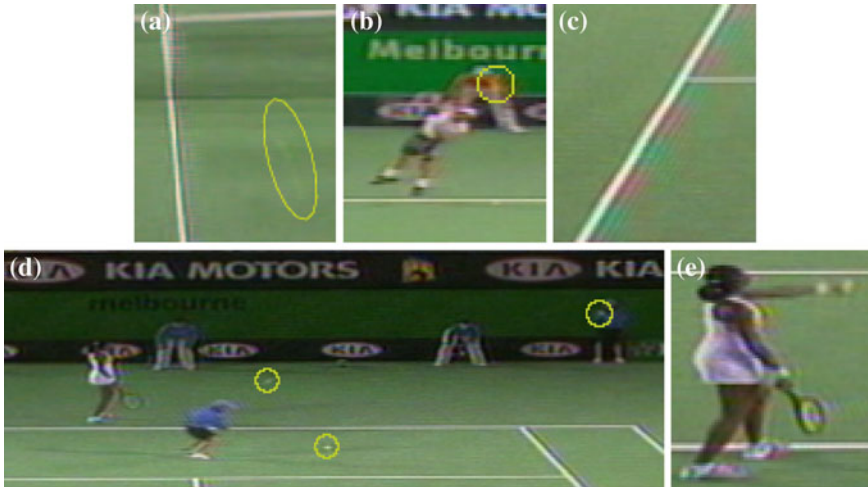
Much of the effort in sports video annotation has been devoted to court games such as tennis and badminton, not only due to their popularity, but also to the fact that court games have well-structured rules. In court games, ball trajectories carry rich semantic information for the annotation. However, tracking a ball in broadcast video is an extremely challenging task. In broadcast videos, the ball can occupy as few as only 5 pixels; it can travel at very high speed and blur into the background; the

---

F. Yan (✉) · W. Christmas · J. Kittler  
Centre for Vision, Speech and Signal Processing, University of Surrey,  
Guildford GU2 7XH, UK  
e-mail: f.yan@surrey.ac.uk

W. Christmas  
e-mail: w.christmas@surrey.ac.uk

J. Kittler  
e-mail: j.kittler@surrey.ac.uk



**Fig. 2.1** Image patches cropped from frames demonstrating the challenges of tennis ball tracking. **a** Ball travels at high speed and blurs into background, making it very hard to detect. **b** Far player serves. The ball region contains only a few pixels, and due to low bandwidth of colour channel, 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** 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. **e** A wristband can look very similar to the ball, and can form a smooth trajectory as the player strikes the ball

ball is also subject to occlusion and sudden change of motion direction. Furthermore, motion blur, occlusion, and abrupt motion change tend to occur together, when the ball is close to one of the players. Example images demonstrating the challenges in tennis ball tracking are shown in Fig. 2.1.

As a result of these challenges, many existing tennis annotation systems avoid ball tracking and rely only on audio and player information [6, 10, 12, 15, 31]. As a result, they can only annotate at a crude level, e.g. highlight detection [10], shot type classification [13], action recognition [31]. In this chapter, we explicitly address the challenges, and propose a fully automatic tennis ball tracking algorithm. Before presenting the algorithm, in the following we briefly review related work in the literature.

### 2.1.1 Related Work

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. Depending on the way, these two pieces of

information are combined, tracking algorithms can be broadly categorised into track-after-detection (TAD) type of approach and track-before-detection (TBD) type.<sup>1</sup>

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. The tracking problem then involves two elements: data association and estimation. While 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); estimation deals with measurement inaccuracy, i.e. what is the “true state” of the object, 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 and defence applications [1, 2, 17, 19, 24], 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 [11, 21, 28]

A tennis ball is a small and fast-moving object with few features. We therefore believe the TAD approach is more suitable for tracking a tennis ball. 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. 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.

Several tennis ball tracking algorithms have been reported in the literature [16, 18, 22, 23, 25, 30]. All these algorithms adopt the TAD approach. Some of the existing algorithms have been successfully applied in practice for enhanced broadcast [23, 25]. In [23, 25], multiple cameras are used. This allows tracking the ball in the 3D real world space, which significantly simplifies the data association problem.

Techniques in [16, 18, 22, 30], on the other hand, rely on a single camera. In this sense, they are more closely related to the scope of this chapter. In [18, 22, 30], the focus is on ball candidate generation rather than data association. Typically, 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

---

<sup>1</sup> Note that TBD is short for track-before-detection instead of track-by-detection, which is in fact TAD (track-after-detection).

with a nearest neighbour type of single hypothesis scheme. This implicitly imposes the assumption that the false candidate rate is very low, and the true ball detection rate is reasonably high.

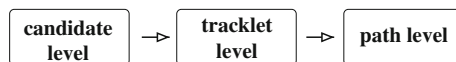
In [16], the authors propose a data association algorithm under the name of Robust Data Association (RDA), and use RDA to track a tennis ball in monocular sequences. RDA does not use the naive single hypothesis data association scheme. The key idea of RDA is to treat data association as a motion model 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.

The remainder of this chapter is organised as follows. In Sect. 2.2 we propose a layered data association (LDA) algorithm for tennis ball tracking. Experimental evaluation is presented and discussed in Sect. 2.3. Finally, Sect. 2.4 concludes the chapter.

## 2.2 Layered Data Association

In this section, we present our three-layered data association scheme for tracking the tennis ball. From bottom to top, the three layers are: candidate level association, tracklet level association and path level analysis (Fig. 2.2).

1. Assume ball candidates in each frame have been generated. Also, assume a temporal sliding window is moved over the sequence. At the **candidate level**, 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 fit a dynamic model to the three candidates. 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 a “tracklet”.
2. As the sliding window moves, a sequence of tracklets is generated. These tracklets may have originated from tennis balls or from clutter. 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 of



**Fig. 2.2** The three layers of the proposed data association scheme

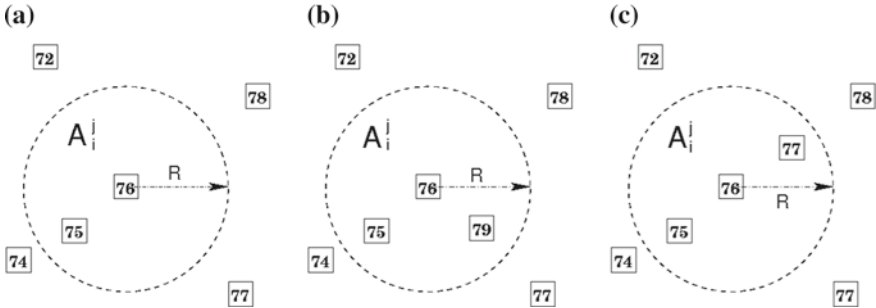
this ball were already known, the shortest path between the two tracklets (nodes) in the graph would then correspond to the trajectory of the ball.

3. We relax above 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.

### 2.2.1 Candidate Level Association

Assume a tennis sequence is composed of  $K$  frames 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  $k$ ,  $c_k^j$  is the  $j$ th of them, and  $k = 1, \dots, K$ . Assume a temporal sliding window containing  $2V + 1$  frames is moved over the sequence. 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  $\mathcal{C}^{(i)} = \{\mathcal{C}_{i-V}, \dots, \mathcal{C}_{i+V}\}$  onto the row-column image plane. In this image plane, centred at each  $c_i^j \in \mathcal{C}_i$  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 if at least one candidate from  $\mathcal{C}_{i-1}$  and at least one candidate from  $\mathcal{C}_{i+1}$  fall into  $A_i^j$  (see Fig. 2.3). Assume  $c_{i-1}^{j'} \in \mathcal{C}_{i-1}$  and  $c_{i+1}^{j''} \in \mathcal{C}_{i+1}$  are found inside  $A_i^j$ , throughout the rest of this chapter, we call the three candidates  $c_{i-1}^{j'}$ ,  $c_i^j$  and  $c_{i+1}^{j''}$  a seed triplet.

The acceleration of a tennis ball in the 3D world space is constant if the air resistance is ignored, since the ball is subject only to gravity. According to perspective geometry [9], after the projective transformation to the 2D image plane, the motion of the ball is approximately constant acceleration, as long as  $\Delta Z$  is negligible compared



**Fig. 2.3** 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, b** For the candidate in frame 76, no seed triplet is found. **c** Sufficient candidates are found in the circular area to form a seed triplet: one from  $\mathcal{C}_{75}$  and one from  $\mathcal{C}_{77}$

to  $Z$ , where  $\Delta Z$  is the distance the ball moves in the world space between two frames along the direction perpendicular to the image plane, and  $Z$  is the distance from the ball to the camera along the same direction.

Consider any three candidates detected in frames  $k_1$ ,  $k_2$  and  $k_3$ , where  $k_1 < k_2 < k_3$ . Let the positions of the candidates in the image plane be  $\mathbf{p}_1$ ,  $\mathbf{p}_2$  and  $\mathbf{p}_3$ , respectively. A constant acceleration model  $M$  can be solved as:

$$\mathbf{v}_1 = \frac{\mathbf{p}_2 - \mathbf{p}_1}{\Delta k_{21}} - \frac{\Delta k_{21} \times \mathbf{a}}{2} \quad (2.1)$$

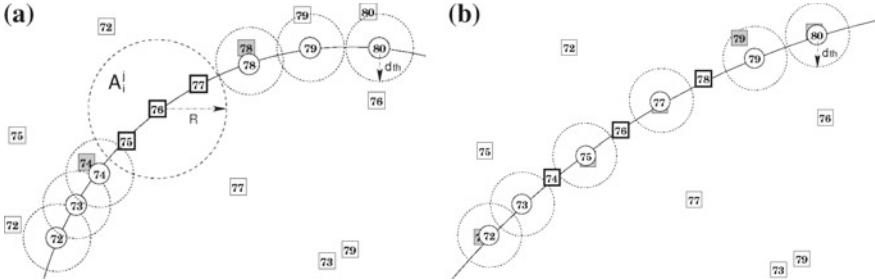
$$\mathbf{a} = 2 \times \frac{\Delta k_{21} \times (\mathbf{p}_3 - \mathbf{p}_2) - \Delta k_{32} \times (\mathbf{p}_2 - \mathbf{p}_1)}{\Delta k_{21} \times \Delta k_{32} \times (\Delta k_{21} + \Delta k_{32})} \quad (2.2)$$

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

$$\hat{\mathbf{p}}_k = \mathbf{p}_1 + \Delta k \times \mathbf{v}_1 + \frac{(\Delta k)^2}{2} \times \mathbf{a} \quad (2.3)$$

where  $\Delta k = k - k_1$ . Such a model can be fitted to any three candidates detected in different frames. We apply it to the seed triplet found inside  $A_i^j$ , as illustrated in Fig. 2.4a. In this special case,  $k_1 = i - 1$ ,  $k_2 = i$  and  $k_3 = i + 1$ .

Compared to the random sampling scheme in Robust Data Association (RDA) [16], the advantage of using seed triplets for model fitting is that a seed triplet



**Fig. 2.4** Model fitting and model optimising. *Squares with numbers* candidates detected in different frames, including true positives and false positives. *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. **a** Fitting a motion model to the seed triplet in Fig. 2.3c. **b** Optimising the fitted model. Assume in **a** 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.  $\hat{k} = 74$ ,  $\hat{k} = 78$  and  $\hat{k} = 76$ . The new triplet (*bold squares in b*) is then used to compute a “better” model

has a higher probability of containing only true positives than a sample set that is drawn randomly from all candidates in the sliding window [20]. 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 Fig. 2.4a. 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 by recursively optimising the model using supports found in the previous iteration [4]. First, we define the support of a model as a candidate that is consistent with the model. More precisely, a candidate  $c_k^j \in \mathcal{C}^{(i)}$  located at  $\mathbf{p}_k^j$  is said to be a support if  $d(\hat{\mathbf{p}}_k, \mathbf{p}_k^j) < d_{\text{th}}$ , where  $d(\cdot, \cdot)$  is the Euclidean distance between two points;  $\hat{\mathbf{p}}_k$  is the estimated ball position at time  $k$  as given by the model; and  $d_{\text{th}}$  is a predefined threshold. In the rare case where more than one candidate satisfies this condition at time  $k$ , only the one with the smallest  $d(\hat{\mathbf{p}}_k, \mathbf{p}_k^j)$  is selected as a support.

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

$$\begin{aligned} \dot{k} &= \min k & \forall c_k^j \in \mathcal{S} \\ \ddot{k} &= \max k & \forall c_k^j \in \mathcal{S} \\ \ddot{k} &= \arg \min_k ||\ddot{k} - k| - |k - \dot{k}|| & \forall c_k^j \in \mathcal{S} \end{aligned} \quad (2.4)$$

We use the three candidates in  $\mathcal{S}$  from frame  $\dot{k}$ ,  $\ddot{k}$  and  $\ddot{k}$  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. 2.4b).

Now, we need a measure of the quality of a model. RANSAC-like algorithms normally use the number of supports a model gets. In [27], a maximum likelihood version of RANSAC, MLESAC, is proposed as a more accurate measure. However, MLESAC involves estimation of mixture parameters of a likelihood function [16, 26], which can be complicated and computationally expensive. In our implementation, the following cost function [27] is adopted:

$$C = \sum_{k=i-V}^{i+V} \sum_j \rho(\mathbf{p}_k^j) \quad (2.5)$$

where

$$\rho(\mathbf{p}_k^j) = \begin{cases} d^2(\hat{\mathbf{p}}_k, \mathbf{p}_k^j) & \text{if } d(\hat{\mathbf{p}}_k, \mathbf{p}_k^j) < d_{\text{th}} \\ d_{\text{th}}^2 & \text{if } d(\hat{\mathbf{p}}_k, \mathbf{p}_k^j) \geq d_{\text{th}} \end{cases} \quad (2.6)$$

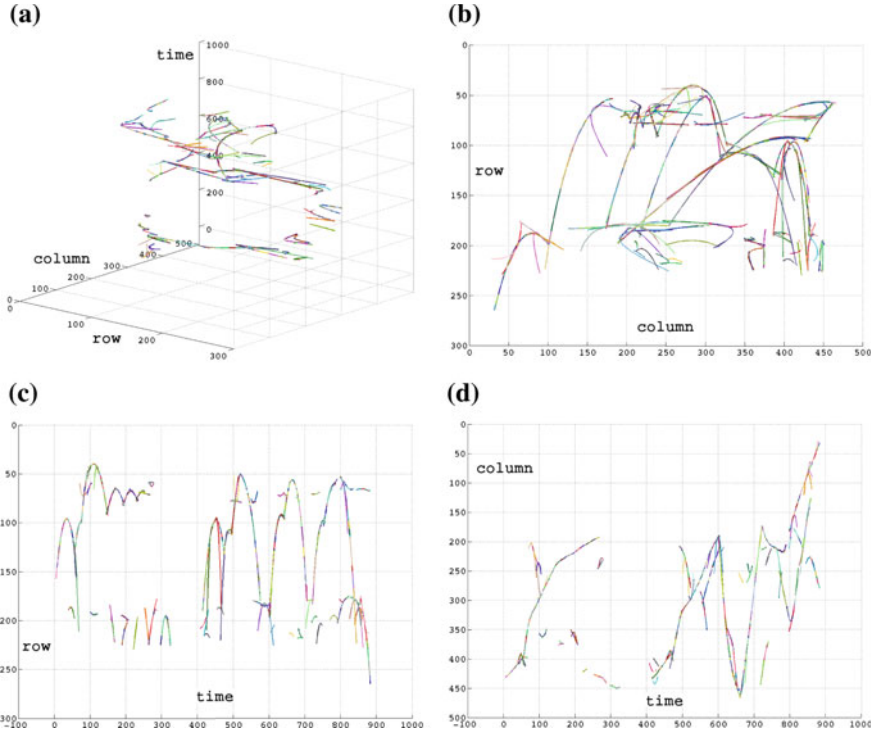
and a smaller  $C$  indicates a better model.

Having defined  $C$ , the optimisation loop terminates when the support set  $\mathcal{S}$  stops expanding, or when  $C$  increases. More specifically, let  $M^{(z)}$  be the model after the  $z$ th iteration,  $C^{(z)}$  be its cost,  $\dot{k}^{(z)}$  and  $\ddot{k}^{(z)}$  be defined as in Eq. (2.4). The optimisation loop terminates if

$$(\dot{k}^{(z+1)} = \dot{k}^{(z)} \wedge \ddot{k}^{(z+1)} = \ddot{k}^{(z)}) \vee (C^{(z+1)} > C^{(z)}) \quad (2.7)$$

and  $M^{(z)}$  is retained as the final fitted model to the current seed triplet.

A tracklet  $T$  is defined as the combination of a parameterised model  $M$  and its support set  $\mathcal{S}$ , i.e.  $T = \{M, \mathcal{S}\}$ . For interval  $I_i$ , the above model-fitting process is applied to each seed triplet that contains each  $c_i^j$  in  $\mathcal{C}_i$ . We then retain only tracklets that have more than  $m_{\text{th}}$  supports, and denote the  $l$ th retained tracklet in  $I_i$  by  $T_i^l = \{M_i^l, \mathcal{S}_i^l\}$ . As the sliding window moves, a sequence of tracklets is generated. Figure 2.5 plots all 672 tracklets generated in an example sequence in the row-column-time 3D space. In the figure, a tracklet  $T_i^l$  is plotted as its dynamic model  $M_i^l$ , and is bounded by  $\dot{k}_i^l$  and  $\ddot{k}_i^l$ .



**Fig. 2.5** All the generated tracklets in an example sequence plotted in random colours. **a** 3D view. **b** Projection on the row-column plane. **c** Projection on the row-time plane. **d** Projection on the column-time plane



### 2.2.2 Tracklet Level Association

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

$$\mathcal{N} = \{\mathcal{N}_{1+V}, \mathcal{N}_{2+V}, \dots, \mathcal{N}_{K-V-1}, \mathcal{N}_{K-V}\} \quad (2.8)$$

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  $n_u^p$  to node  $n_v^q$ ,  $e_{u,v}^{p,q}$ , exists in  $\mathcal{E}$ , if

$$u < v \quad \wedge \quad \dot{k}_v^q - \ddot{k}_u^p \leq k_{\text{th}} \quad (2.9)$$

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  $k_{\text{th}}$  successive frames, where  $k_{\text{th}} \in \mathcal{N}^+$ . An example of graph topology is given in Fig. 2.6.

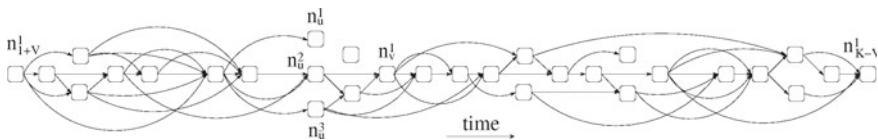
Both  $M$  and  $\mathcal{S}$  of the tracklets are used to define the edge weight between two nodes. First, two nodes  $n_u^p$  and  $n_v^q$  are said to be “overlapping” if

$$u < v \quad \wedge \quad \dot{k}_v^q - \ddot{k}_u^p \leq 0 \quad (2.10)$$

For overlapping nodes, their support sets are used. Two overlapping nodes are “conflicting” (not compatible) if for  $\check{k}_v^q \leq k \leq \check{k}_u^p$ ,  $\exists k$  such that

$$\begin{aligned}
& (\exists c_k' \in \mathcal{S}_u^p \wedge \nexists c_k'' \in \mathcal{S}_v^q) \vee (\nexists c_k' \in \mathcal{S}_u^p \wedge \exists c_k'' \in \mathcal{S}_v^q) \\
& \vee (\exists c_k' \in \mathcal{S}_u^p \wedge \exists c_k'' \in \mathcal{S}_v^q \wedge \mathbf{p}_k' \neq \mathbf{p}_k'') \quad (2.11)
\end{aligned}$$

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



**Fig. 2.6** 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  $10^2$ – $10^3$ . Far fewer nodes are plotted in this figure for ease of visualisation

$$w_{u,v}^{p,q} = \begin{cases} \infty & \text{if } n_u^p \text{ and } n_v^q \text{ are conflicting} \\ 0 & \text{otherwise} \end{cases} \quad (2.12)$$

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

$$w_{u,v}^{p,q} = \min d(\hat{\mathbf{p}}_{u,k}^p, \hat{\mathbf{p}}_{v,k}^q), \quad \forall k \in [\ddot{k}_u^p, \ddot{k}_v^q] \quad (2.13)$$

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. 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.

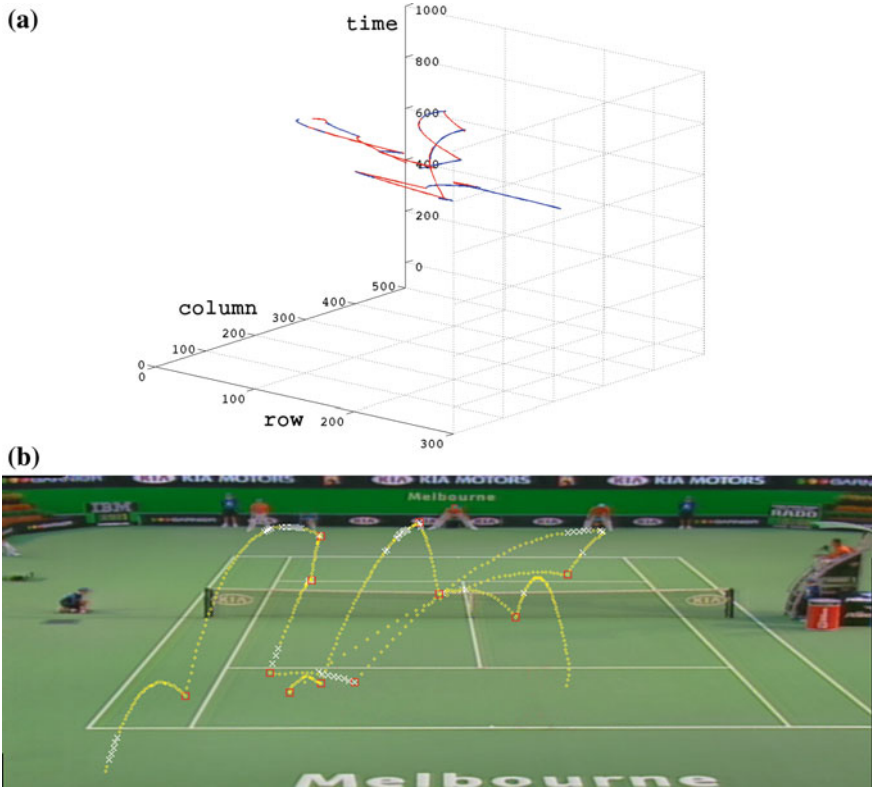
The edge weight defined in Eq. (2.12) and Eq. (2.13) can be thought of as a compatibility measure. For now, we assume that there is only one ball to track, and the first and last nodes that have originated from this ball are already known. The Dijkstra’s algorithm [8] can then be applied to find the shortest path between the pair of nodes. We manually specify the first and last ball-originated nodes in the third play of the example sequence as source node and destination node. The result of applying Dijkstra’s algorithm is illustrated in Fig. 2.7: a shortest path that corresponds to the ball trajectory. According to the definition of edge weight, 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.

In tennis ball tracking, the points at which the ball changes its motion abruptly correspond to key events such as hit and bounce, and provide important information for high-level annotation. We use the algorithm of generalised edge-preserving signal smoothing to detect these key events. A more detailed description can be found in our previous work [29].

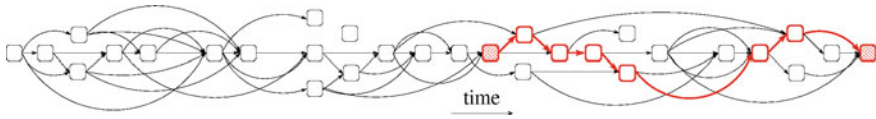
### 2.2.3 Path Level Analysis

In this section, we relax the assumptions made in the previous section, and show how this relaxation 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.

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 any pair of nodes that has 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 the paths that correspond to ball trajectories form a subset of  $\mathcal{Q}$ . Our goal is to reduce the original set of APSP  $\mathcal{Q}$  to its subset that contains only paths that correspond to the ball trajectories (Fig. 2.8).



**Fig. 2.7** **a** Shortest path given by Dijkstra’s algorithm. Adjacent nodes in the *shortest* path are plotted alternatively in blue and red. 3D view. **b** 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



**Fig. 2.8** An illustration of the *shortest* path in Fig. 2.7 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

To this end, we need to define the relative quality and compatibility of the paths. Recall that the weight of a path is the sum of the weights of all edges the path goes through. We then define the length of a path to be 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”. However, we chose to use “shortest path” for the sake of consistency with the terminology used in the literature.

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:

$$P_1 \begin{cases} > \\ = \\ < \end{cases} P_2 \quad \text{if } (W_1 - W_2) \begin{cases} < \\ = \\ > \end{cases} \alpha \cdot (L_1 - L_2) \quad (2.14)$$

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  $\alpha$  is a controllable parameter with a 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  $P_2$ . It easily follows that the set  $\mathcal{Q}$  equipped with an operator “ $\geq$ ” satisfies transitivity, antisymmetry and totality. According to order theory [7],  $\mathcal{Q}$  associated with operator “ $\geq$ ” is a totally ordered set. The relative quality of the paths is defined.

The definition of pair-wise compatibility of the paths is straightforward. Under the assumption that two objects cannot produce the same candidate in a frame, 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.

---

**Algorithm 1** The path reduction algorithm

---

**Input:** Set  $\mathcal{P}$  with  $p$  paths, and empty set  $\mathcal{B}$ .

**Output:** A reduced set  $\mathcal{P}$ .

```

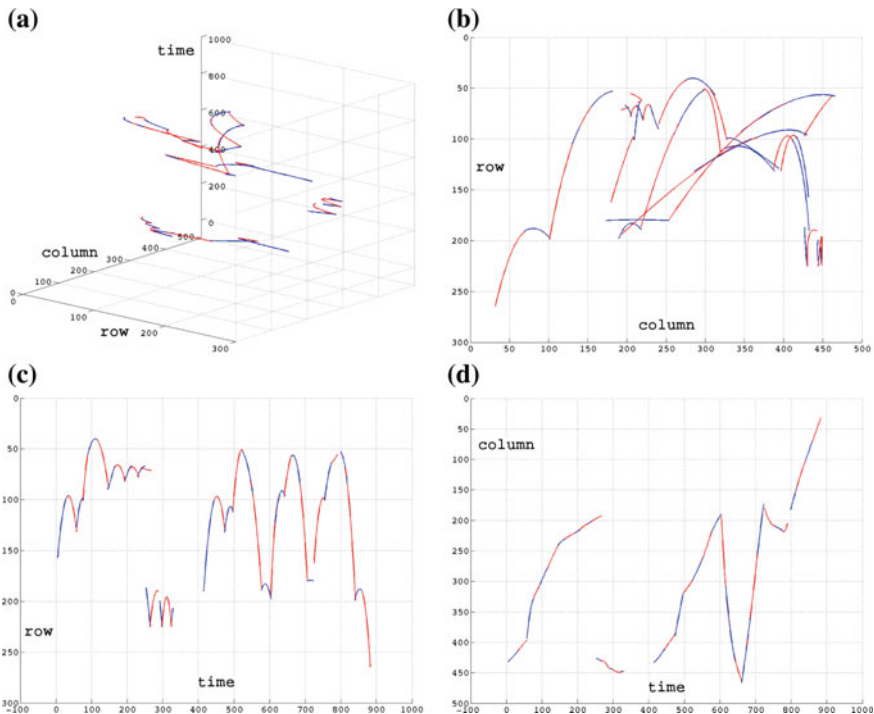
1: while  $\mathcal{P}$  is not empty do
2:   Remove the best path  $P^*$  in  $\mathcal{P}$  from  $\mathcal{P}$ 
3:   if  $P^*$  is compatible with all paths in  $\mathcal{B}$  then
4:     add  $P^*$  to  $\mathcal{B}$ 
5:   end if
6: end while

```

---

We propose a simple path reduction algorithm (see Algorithm 1) to prune the APSP set. According to the definition of relative quality and pair-wise compatibility of the paths, the paths in the resulting subset are mutually compatible, and are optimal in the sense that the best remaining path in  $\mathcal{P}$  was always considered first. We call the output of the path reduction algorithm  $\mathcal{B}$  the Best Set of Compatible Paths (BSCP).

Now, we have a complete data association algorithm for tracking multiple objects fully automatically. We apply the complete algorithm to the example sequence. 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  $Q = 87,961$  paths is obtained. The path reduction algorithm is then applied, which gives a BSCP  $\mathcal{B}$  containing 11 paths. In descending order, the numbers of supports (lengths) of

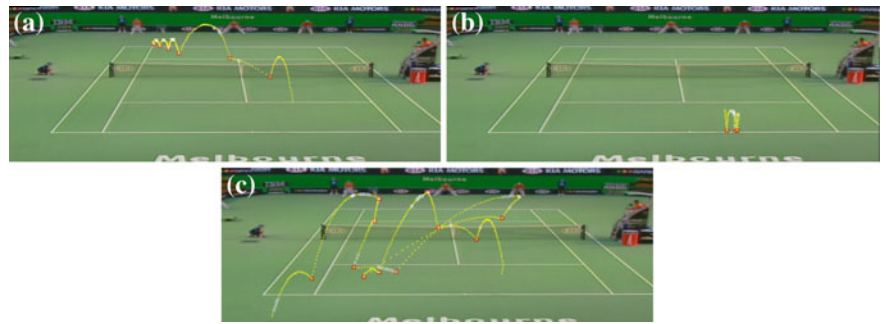


**Fig. 2.9** Results of applying the complete algorithm to the example sequence: three paths in the thresholded BSCP  $\mathcal{B}_{th}$ . Adjacent nodes in each path are plotted alternatively in *blue* and *red*. **a** 3D view. **b** Projection on the row-column plane. **c** Projection on the row-time plane. **d** Projection on the column-time plane

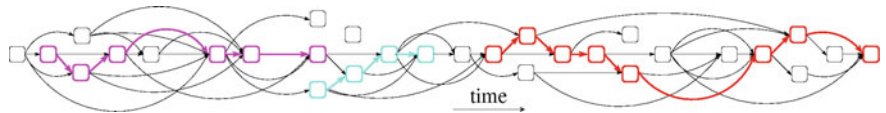
the paths in  $\mathcal{B}$  are: 411, 247, 62, 23, 20, 17, 17, 16, 15, 10, 9. It is a reasonable assumption that a path corresponding to a ball trajectory has more supports than a path corresponding to the motion of a non-ball object, e.g. a wristband. We set a threshold  $L_{th}$  and keep only the paths that have more supports than  $L_{th}$ . This results in a thresholded BSCP  $\mathcal{B}_{th}$  with three 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. 2.9. In Fig. 2.10, the three reconstructed trajectories are superimposed on mosaic images. An illustration of the three paths in the graph is shown in Fig. 2.11.

## 2.3 Experiments

In this section, we first introduce data sets used in our evaluation and tracking algorithms under comparison. We then define performance metrics. Finally, experimental results are presented, including a study of sensitivity of the proposed method to parameters.



**Fig. 2.10** 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 16–260; the second frame 254–321; and the third frame 427–887



**Fig. 2.11** 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

2.3.1 Experimental Data

Experiments were carried out on broadcast tennis videos from three tennis tournaments. A broadcast tennis video is composed of shots, such as play, close-up, crowd, advertisement. A play shot is a shot in which the ball is in play. In a play shot, the camera is usually from behind one of the players, and has small pan, tilt and zoom (PTZ) motion. Tennis ball tracking is performed only on play shots. In this chapter, we also refer to a play shot as a “sequence”.

The three data sets used contain 165 sequences in total. A set with 25 sequences was used as training set, and the other two with 70 sequences each as test sets. Some statistics of the three data sets are shown in Table 2.1. In Table 2.1,  $\bar{N}$  is the average number of false candidates in each frame, and  $r_d$  is the probability that a ball is detected as a candidate (ball detection rate). In our experiments, the parameters of the proposed method were assumed independent of each other, and were tuned

**Table 2.1** Statistics of experimental data

	Game	Court type	No. of seq	No. of frames	Length (min)	$\bar{N}$	$r_d$ (%)
Training set	Australian’03 Women’s	Artificial	25	15,716	5.2	12.1	90.3
Test set 1	Australian’06 Men’s	Artificial	70	60,094	20.0	10.6	91.6
Test set 2	Wimbledon’05 Men’s	Grass	70	43,706	14.6	16.5	88.2

separately on the training set to maximise the F-measure of event detection. The definition of F-measure which will be given shortly in Sect. 2.3.3. The optimal set of parameters obtained during the training was:  $V = 10$ ,  $R = 20.0$ ,  $d_{th} = 5.0$ ,  $m_{th} = 6$ ,  $k_{th} = 20$ ,  $L_{th} = 45$ , and  $\alpha = 1.0$ .

The proposed data association algorithm takes as input the tennis ball candidates in each frame. In each sequence, image frames were first de-interlaced into fields. For the sake of consistency with the terminology used in the literature, we have used “frame” to refer to “field”. After de-interlacing, homographies between frames were calculated by tracking corresponding corners that are automatically detected, and were used to compensate the global motion (camera PTZ) between frames [3, 14]. Foreground moving objects were then extracted by differencing nearby frames and thresholding the differences. Finally, a simple filter was applied to keep only foreground blobs with an appropriate size. These blobs were used as tennis ball candidates for tracking algorithms.

### 2.3.2 Algorithms Under Comparison

For comparison with the proposed algorithm, PDA [1] and RDA [16] were also implemented. Due to the abrupt motion changes, PDA lost track in most sequences. We will therefore focus on the comparison between the proposed method and RDA. In RDA, the number of trials,  $n$ , is chosen so that the probability of finding a set consisting entirely of true positives,  $P$ , is greater than a threshold  $P_0$ . It has been shown [5] that

$$n \geq \log(1 - P_0) / \log(1 - \varepsilon^s) \quad (2.15)$$

where  $\varepsilon$  is the ratio of the number of true positives to the number of all candidates, and  $s$  is the size of each sample set. In our experiments,  $P_0$  was set to 0.95, and the interval size was set to 15, both as suggested in [16]. According to Eq. (2.15), at least 8,161 trials were needed. The mixture parameters of the likelihood function in RDA were estimated recursively until convergence, also as suggested in [16]. As to dynamic model, the same constant acceleration model as defined in Eq. (2.1) and Eq. (2.2) was used. Since RDA cannot deal with track initiation and track termination, we manually specified the frames in which it was applied.

### 2.3.3 Evaluation Methods

#### 2.3.3.1 Matching the Tracked Trajectories and the Ground Truth Trajectories Automatically

For each sequence, 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. Each trajectory is a sequence of points in

the row-column-time 3D space. Define an assignment scheme as a correspondence between the two sets of trajectories that satisfies: each trajectory in  $\mathcal{T}_{\text{gt}}$  corresponds to at most one trajectory in  $\mathcal{T}_{\text{tr}}$ , and vice versa.

Observe that an assignment scheme consists of correspondence pairs. For a given pair, let  $\mathcal{J}^*$  be the time interval where the two trajectories overlap. For each frame in  $\mathcal{J}^*$ , two ball positions are given, one by the tracker and one by the ground truth. The two positions are compared. If the distance between them 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. Finally, the quality of an assignment scheme is defined as the sum of match scores of all its correspondence pairs. Among all possible assignment schemes, the one with highest quality is chosen.

### 2.3.3.2 Quality of Track Initiation and Track Termination

Having obtained the correspondence between  $\mathcal{T}_{\text{gt}}$  and  $\mathcal{T}_{\text{tr}}$ , we can measure the performance of the tracker from several aspects. Let  $T_{\text{gt}}^i \in \mathcal{T}_{\text{gt}}$  and  $T_{\text{tr}}^i \in \mathcal{T}_{\text{tr}}$  be the trajectories in the  $i$ th pair, and  $\mathcal{J}_{\text{gt}}^i$  and  $\mathcal{J}_{\text{tr}}^i$  be the time interval of  $T_{\text{gt}}^i$  and  $T_{\text{tr}}^i$ , respectively. Moreover, let  $T_{\text{gt}}^j$  be the  $j$ th trajectory in  $\mathcal{T}_{\text{gt}}$  that is not in any correspondence pair, and  $\mathcal{J}_{\text{gt}}^j$  be its time interval; and let  $T_{\text{tr}}^k$  be the  $k$ th trajectory in  $\mathcal{T}_{\text{tr}}$  that is not in any correspondence pair, and  $\mathcal{J}_{\text{tr}}^k$  be its time interval. Now, define

$$I^\cap = \sum_i |\mathcal{J}_{\text{gt}}^i \cap \mathcal{J}_{\text{tr}}^i| \quad (2.16)$$

and

$$I^\cup = \sum_i |\mathcal{J}_{\text{gt}}^i \cup \mathcal{J}_{\text{tr}}^i| + \sum_j |\mathcal{J}_{\text{gt}}^j| + \sum_k |\mathcal{J}_{\text{tr}}^k| \quad (2.17)$$

The quality of track initiation and track termination on this sequence is then:

$$\beta = I^\cap / I^\cup \quad (2.18)$$

where  $\beta$  ranges between 0 and 1. When  $\beta = 1$ , track initiation/termination is said to be perfect.

### 2.3.3.3 Proportion of LOT

In addition to the quality of track initiation and termination, we would also like to know how close the positions given by the tracker are to the ground truth. Consider the  $i$ th pair of trajectories in the best assignment scheme. Two ball positions are



given in each frame in  $\mathcal{S}_{\text{tr}}^i \cap \mathcal{S}_{\text{gt}}^i$ , one by the tracker and the other by the ground truth. We define tracking error as the Euclidean distance between the two positions. For a given sequence, the number of such errors is  $I^\cap$ . A loss-of-track (LOT) is then defined as when tracking error is greater than 6 pixels. Let the number of LOTs in the given sequence be  $I_m^L$ . The proportion of LOT is then defined as:

$$\gamma = I^L / I^\cap \quad (2.19)$$

### 2.3.3.4 Quality of Event Detection

We also measure the performance of the proposed algorithm in terms of quality of event detection. For each trajectory in  $\mathcal{T}_{\text{gt}}$ , points that correspond to motion discontinuities are marked up as ground truth events. These events correspond to racket hitting the ball, the ball bouncing on the ground and the ball touching the net. We then compare the events detected by an event detection algorithm against the ones in ground truth. A detected event in a tracked trajectory  $T_{\text{tr}}^i$  is regarded as “matched” if it is within 3 frames and 5 pixels of a ground truth event in the corresponding ground truth trajectory  $T_{\text{gt}}^i$ . We report the F-measure of event detection, which is the harmonic mean of the precision and the recall. The F-measure ranges from 0 to 1, and a larger value indicates a better performance.

The quality of event detection is obviously affected by the event detection algorithm used, and thus not a direct quality measure of the ball tracking algorithm. However, event detection plays a crucial role in high-level annotation. We present the quality of event detection to give an idea of the performance of the ball tracking algorithm in the context of tennis video annotation.

## 2.3.4 Results and Discussion

Experimental results on the two test sets are shown in Table 2.2. The quality of track initiation/termination of the two methods is not comparable, since in RDA this was manually dealt with. In terms of proportion of LOT and F-measure of event detection, the proposed method outperforms RDA significantly on both test sets. In RDA, estimates are given by the best models in corresponding intervals independently of

**Table 2.2** Experimental results on the test sets

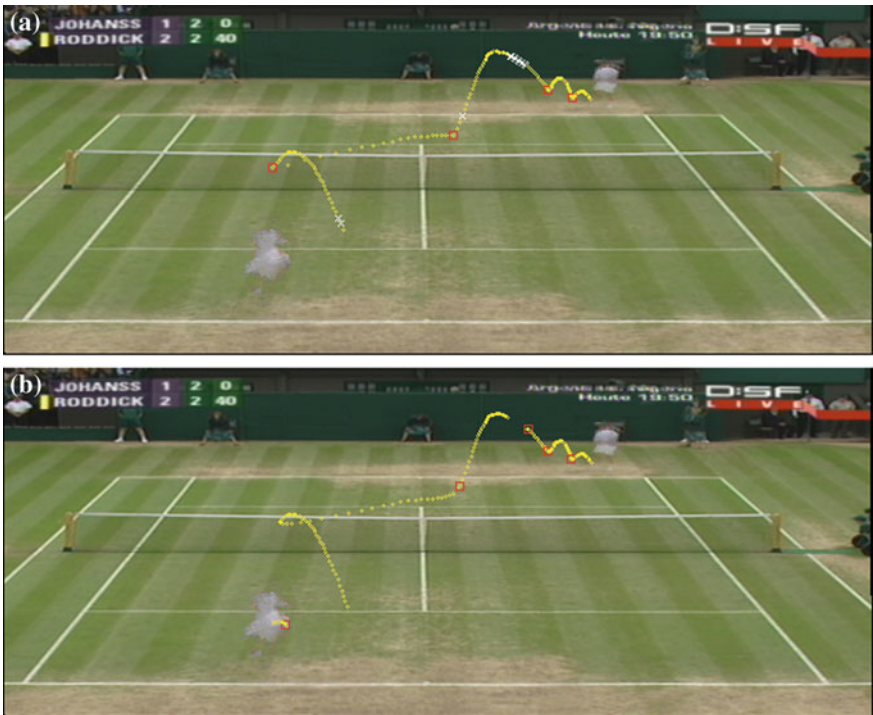
		$\beta$	$\gamma$ (%)	F-measure	Frames/s
Test set 1	Proposed	<b>83.7%</b>	<b>3.51</b>	<b>0.758</b>	<b>98.4</b>
	RDA	—	11.74	0.423	3.6
Test set 2	Proposed	<b>82.2%</b>	<b>3.66</b>	<b>0.729</b>	<b>109.3</b>
	RDA	—	13.23	0.411	3.2

each other, and the concept of “tracklet” does not exist. This significantly limits RDA’s ability to deal with complexities in broadcast tennis video. In fact, RDA works only under a strong assumption: a model fitted to three candidates that have all originated from the ball being tracked is always “better” than that fitted to candidates that have not. This assumption is not true in broadcast tennis sequences because:

1. Ball trajectories can overlap in time, i.e. there may be multiple balls in the same part of a sequence;
2. Clutter-originated candidates can also form smooth trajectories that can be explained by constant acceleration models;
3. A ball can change its motion drastically. As a result, even a model fitted to three ball-originated candidates can have a poor quality.

By contrast, the proposed method slices the data association problem into layers, and each layer is designed to solve the data association problem at its own level. It is therefore more flexible in dealing with the complexities in broadcast tennis video. (see Fig. 2.12 for an example).

The proposed algorithm also has the advantage of being more efficient. The fact that it always starts model fitting from a seed triplet allows it to eliminate false



**Fig. 2.12** The assumption in order for RDA to work is invalid in broadcast tennis video. **a** Tracking results of the proposed method. **b** Tracking results of RDA

**Table 2.3** Sensitivity to parameters

	Opt. param.	RDA	V		R		$d_{th}$		$m_{th}$	
			7	13	15.0	25.0	4.0	6.0	4	8
$\beta$	83.7 %	–	82.8 %	82.9 %	83.6 %	83.7 %	82.9 %	83.0 %	83.0 %	75.9 %
$\gamma$	3.51 %	11.74 %	3.94 %	3.78 %	3.58 %	3.50 %	3.87 %	3.29 %	3.82 %	3.75 %
F-measure	0.758	0.423	0.725	0.770	0.754	0.759	0.732	0.768	0.763	0.696
Frames/s	98.4	3.6	102.2	93.8	100.3	96.6	128.4	81.4	55.7	152.9
	$k_{th}$		$L_{th}$		$\alpha$					
	15	25	30	60	0.001	0.01	0.1	10.0	100.0	1,000.0
$\beta$	76.5 %	85.1 %	80.0 %	84.0 %	42.5 %	49.3 %	78.1 %	76.8 %	75.2 %	75.1 %
$\gamma$	2.78 %	4.46 %	2.89 %	3.29 %	2.10 %	2.14 %	2.64 %	5.78 %	7.48 %	7.60 %
F-measure	0.740	0.747	0.747	0.759	0.538	0.584	0.738	0.700	0.668	0.667
Frames/s	105.6	89.4	92.9	102.4	88.8	91.3	96.0	98.5	100.0	100.2

candidates very quickly. Taking test set 1, for example at each time step, on average 10.6 candidates are evaluated. On average, there are 1.02 seed triplets containing each candidate, and it takes 3.7 iterations for model optimisation to converge. The effective number of models evaluated is then  $10.6 \times 1.02 \times 3.7 \approx 40$ . On the other hand, according to Eq. (2.15), at least 8161 models are needed in RDA. The estimation of mixture parameters of the likelihood function in RDA is also time-consuming. Time in Table 2.2 was measured on a single thread of an Intel Xeon 3.0G computer running Linux, and overhead of disk I/O was included. Both algorithms were implemented in C++.

Finally in Table 2.3 we show the sensitivity of the proposed algorithm to parameters using test set 1. It is evident from the table that the proposed algorithm is not sensitive to small changes of most parameters. However, when the parameter for determining the relative quality of two paths,  $\alpha$ , varies over a large range, paths can be inappropriately merged or broken, severely degrading track initiation/termination and event detection. We can also see that the proposed method is sensitive to the increase of the tracklet threshold  $m_{th}$ . This is because when  $m_{th}$  is too high, ball-originated tracklets are discarded, and the shortest path algorithm is forced to take a wrong path. On the other hand, when  $m_{th}$  is low, false tracklets are allowed. However, the shortest path algorithm can remove them effectively.

## 2.4 Conclusions

In this chapter, we have presented a layered data association algorithm with graph-theoretic formulation for tracking multiple objects that undergo switching dynamics in clutter. At the object candidate level, tracklets are grown from sets of candidates that have high probabilities of containing only true positives. The tracklet association

problem is then mapped onto a graph, and solved with an all-pairs shortest path formulation and path level analysis. This results in a fully automatic data association algorithm which can handle multiple object scenarios. The proposed data association algorithm has been applied to tennis sequences from several tennis tournaments to track tennis balls. Comparative experiments show that it works well on sequences where other data association methods perform poorly or fail completely.

## References

1. Bar-Shalom Y, Fortmann TE (1988) Tracking and data association. Academic Press Inc, Boston
2. Bar-Shalom Y, Kirubarajan T (2001) Estimation with applications to tracking and navigation. Wiley, New York
3. Christmas W, Kostin A, Yan F, Kolonias I, Kittler J (2005) A system for the automatic annotation of tennis matches. In: Fourth international workshop on content-based multimedia indexing
4. Chum O, Matas J, Kittler J (2003) Locally optimized Ransac. In: DAGM-symposium, pp 236–243
5. Chum O, Matas J, Obdrzalek S (2004) Enhancing Ransac by generalized model optimization. *Asian Conf Comput Vis* 2:812–817
6. Coldefy F, Boutheimy P, Betser M, Gravier G (2004) Tennis video abstraction from audio and visual cues. In: IEEE international workshop on multimedia signal processing
7. Davey B, Priestley H (2002) Introduction to lattices and order. Cambridge University Press, Cambridge
8. Dijkstra E (1959) A note on two problems in connexion with graphs. *Numer Math* 1:269–271
9. Hartley R, Zisserman A (2004) Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, Cambridge
10. Huang Y, Chiou C, Sandnes F (2009) An intelligent strategy for the automatic detection of highlights in tennis video recordings. *Expert Syst Appl* 36:9907–9918
11. Isard M, Blake A (1998) Condensation—conditional density propagation for visual tracking. *Int J Comput Vis* 1(29):5–28
12. Kijak E, Gravier G, Oisel L, Gros P (2003) Audiovisual integration for tennis broadcast structuring. In: International workshop on content-based multimedia indexing
13. Kijak E, Gravier G, Gros P, Oisel L, Bimbot F (2003) Hmm based structuring of tennis videos using visual and audio cues. *IEEE Int Conf Multimed Expo* 3:309–312
14. Kittler J, Christmas W, Kostin A, Yan F, Kolonias I, Windridge D (2005) A memory architecture and contextual reasoning framework for cognitive vision. In: 14th Scandinavian conference on image analysis
15. Lai J, Chen C, Kao C, Chien S (2011) Tennis video 2.0: a new presentation of sports videos with content separation and rendering. *J Vis Commun Image Represent* 22:271–283
16. Lepetit V, Shahrokhni A, Fua P (2003) Robust data association for online applications. *CVPR* 1:281–288
17. Mazor E, Averbuch A, Bar-Shalom Y, Dayan J (1998) Interacting multiple model methods in target tracking: a survey. *IEEE Trans Aerosp Electron Syst* 34(1):103–122
18. Miyamori H, Iisaku S (2000) Video annotation for content-based retrieval using human behavior analysis and domain knowledge. In: International conference on automatic face and gesture recognition, pp 320–325
19. Musicki D, La Scala BF, Evans RJ (2004) Integrated track splitting filter for manoeuvring targets. In: IEEE international conference on information fusion
20. Myatt DR, Torr PHS, Nasuto SJ, Bishop JM, Craddock R (2002) NAPSAC: high noise, high dimensional Robust estimation—it's in the bag. In: British machine vision conference, pp 458–467

21. Nummiaro K, Merier E, van Gool L (2003) An adaptive color-based particle filter. *J Image Vis Comput* 21(1):99–110
22. Pingali GS, Jean Y, Carlbom I (1998) Real time tracking for enhanced tennis broadcasts. In: *CVPR*, pp 260–265
23. Pingali GS, Opalach A, Jean Y (2000) Ball tracking and virtual replays for innovative tennis broadcasts. In: *ICPR*, pp 4152–4156
24. Streit R, Luginbuhl T (1995) Probabilistic multi-hypothesis tracking. Technical Report
25. The Hawkeye Tennis System (2014) <http://www.hawkeyeinnovations.co.uk>
26. Tordoff BJ, Murray DW (2005) Guided-MLESAC: faster image transform estimation by using matching priors. *PAMI* 27(10):1523–1535
27. Torr PHS, Zisserman A (2000) MLESAC: a new robust estimator with application to estimating image geometry. *Comput Vis Image Underst* 78:138–156
28. Viola P, Jones M (2004) Robust real-time object detection. *IJCV* 57(2):137–154
29. Yan F, Kostin A, Christmas W, Kittler J (2006) A novel data association algorithm for object tracking in clutter with application to tennis video analysis. In: *CVPR*
30. Yu X, Sim C, Wang JR, Cheong L (2004) A trajectory-based ball detection and tracking algorithm in broadcast tennis video. *ICIP* 2:1049–1052
31. Zhu G, Huang Q, Xu C, Xing L (2007) Human behavior analysis for highlight ranking in broadcast racket sports video. *IEEE Trans Multimed* 9(6):1167–1182

Computer Vision in Sports

Moeslund, Th.B.; Thomas, G.; Hilton, A. (Eds.)

2014, XI, 319 p. 171 illus., 139 illus. in color., Hardcover

ISBN: 978-3-319-09395-6