Layered Data Association Using Graph-Theoretic Formulation with Application to Tennis Ball Tracking in Monocular Sequences

Fei Yan, William Christmas, and Josef Kittler, *Member, IEEE,*

***Abstract*—In this paper, we propose a multi-layered data association scheme with graph-theoretic formulation for tracking multiple objects that undergo switching dynamics in clutter. The proposed scheme takes as input object candidates detected in each frame. At the object candidate level, "tracklets" are "grown" from sets of candidates that have high probabilities of containing only true positives. 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 association problem is then formulated as an all-pairs shortest path (APSP) problem in this graph. Finally, at the path level, by analyzing the all-pairs shortest paths, all object trajectories are identified, and track initiation and track termination are automatically dealt with. By exploiting a special topological property of the graph, we have also developed a more efficient APSP algorithm than the general-purpose ones. The proposed data association scheme is applied to tennis sequences to track tennis balls. Experiments show that it works well on sequences where other data association methods perform poorly or fail completely.**

***Index Terms*—Data association, Trajectory reconstruction, Graph theory, Shortest path**

I. Introduction

THE rapid growth of sports video database demands efficient tools for automatic sports video annotation. A sports video annotation system has many potential applications, *e.g.* video retrieval, enhanced broadcast, object-based video encoding, analysis of player tactics. In automatic video annotation, high-level descriptions rely on low-level features. In the context of a tennis game, the evolution of a game is described by key events such as the ball being hit, the ball bouncing on the ground, *etc.* To detect these important events, it is essential to track the ball trajectory.

Let us assume we have a ball candidate generation module, where a ball is detected with a certain probability along with some false alarms. The data association problem, *i.e.* the problem of determining which candidates are ball-originated and which are clutter-originated, is the key problem to solve in tennis ball tracking. Data association has been intensively studied over the past several decades, especially inside the automatic control and aerospace/defense communities. Many algorithms have been developed, ranging in complexity from Track-splitting Filter [1], Probabilistic Data Association (PDA) [2], Viterbi Data Association (VDA) [3], to Probabilistic Multi-Hypothesis Tracker (PMHT) [4], and Multi-Hypothesis Tracker (MHT) [5]. Numerous extensions for tracking multiple objects and maneuvering objects have also been proposed, *e.g.* [6], [7], [8], [9], [10]. However, these "classical" data association algorithms are not suitable for tennis ball tracking.

Firstly, most classical data association techniques are intrinsically iterative. That is, the association/estimation at the current frame relies on that in previous frames. When applying an iterative algorithm to tennis ball tracking, there is a major difficulty: abrupt changes of tennis ball motion. 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 maneuver, which is more or less a solved problem [7], [8], [9], [10]. Most iterative trackers would lose track in such a situation.

Secondly, most classical data association techniques model background clutter as uniform white noise, and require the clutter density to be known *a priori* . However, in tennis video, false ball candidates can often come from the players, the rackets, or the advertisement boards. As a result, false candidates show a spatially clustered pattern. False candidates can also be strongly temporally correlated. For example, candidates that have originated from a wristband can form smooth trajectories as the player strikes the ball. All these make the assumption of uniform white noise invalid.

Thirdly, 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

Manuscript received December, 2006; revised August, 2007.

Authors are with the Center for Vision, Speech and Signal Processing, University of Surrey, UK.

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

Fig. 1. Images cropped from frames showing that tennis ball tracking is a challenging task. (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 color is strongly affected by the background color. (c) Chroma noise caused by PAL cross-color 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.

monocular tennis video are shown in Fig. 1.

Finally, most existing data association techniques are online algorithms and have small decision delays. That is, if the association/estimation result in frame i is reported when candidates in frame $i + D$ are received, D is either zero or a small number. This is because these algorithms have been developed for online and near real-time applications. In a post-production situation, however, this requirement is relaxed. Distant "future" information can be used to help decide association at the current frame. Since in theory there is an inverse relationship between the decision delay and the quality of the association, a carefully designed off-line algorithm is expected to outperform any online algorithms.

A. Related Work

Several tennis ball tracking algorithms have been reported in the literature [11], [12], [13], [14], [15]. Some of them have even been successfully used in practice for enhanced broadcast [11], [12]. However, techniques in [11], [12] rely on multiple cameras. On the other hand, most existing tennis ball tracking algorithms for monocular sequences focus on the estimation problem rather than the data association problem [13], [14]. 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 with a Kalman filter. In other words, the data association problem is implicitly addressed with a nearest neighbor or strongest neighbor type of single hypothesis scheme. This inevitably imposes the assumption that the false candidate rate is very low, and the true object detection rate is reasonably high.

In [15], 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 a 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 [16] algorithm is used to find the motion model that is best at explaining the candidates inside the window, *i.e.* the model with maximum likelihood. 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.

RDA is more robust to false candidates than single hypothesis approaches. However, several deficiencies of RDA are noticed. Firstly, RDA is a single-object tracking algorithm, and by itself cannot deal with track initiation/termination. This means RDA is not applicable for tennis sequences that have complex track initiation/termination scenarios of multiple objects. Secondly, like most RANSAC-based algorithms, in RDA, samples are drawn randomly from all candidates in the sliding window. 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 RDA's model-fitting approach to the data association problem, and trying to remedy the weaknesses of RDA, we have developed a layered data association algorithm with graph theoretic formulation for tennis ball tracking in monocular sequences. The proposed scheme has been tested on large data sets that contains tennis sequences from three tournaments. Comparative experiments show that it works well on sequences where other data association methods perform poorly or fail completely.

The rest of this paper is organized as follows: Section II gives an overview of the proposed scheme. Section III to Section V describe the three layers of the proposed scheme: candidate level association, tracklet level association, and path level analysis, respectively. Section VI discusses processing speed considerations. An efficient all-pairs shortest path algorithm is also presented in this section. We then evaluate in Section VII the performance of the proposed algorithm. Finally, conclusions are given in Section VIII.

II. Overview of the Strategy

To better appreciate the ball tracking problem at hand, we plot in Fig. 2 (a) all the ball candidates in a sequence in a column-row-time 3D space. Note that since the camera has pan, tilt and zoom (PTZ) motion, candidate positions in different frames have been projected into a common coordinate

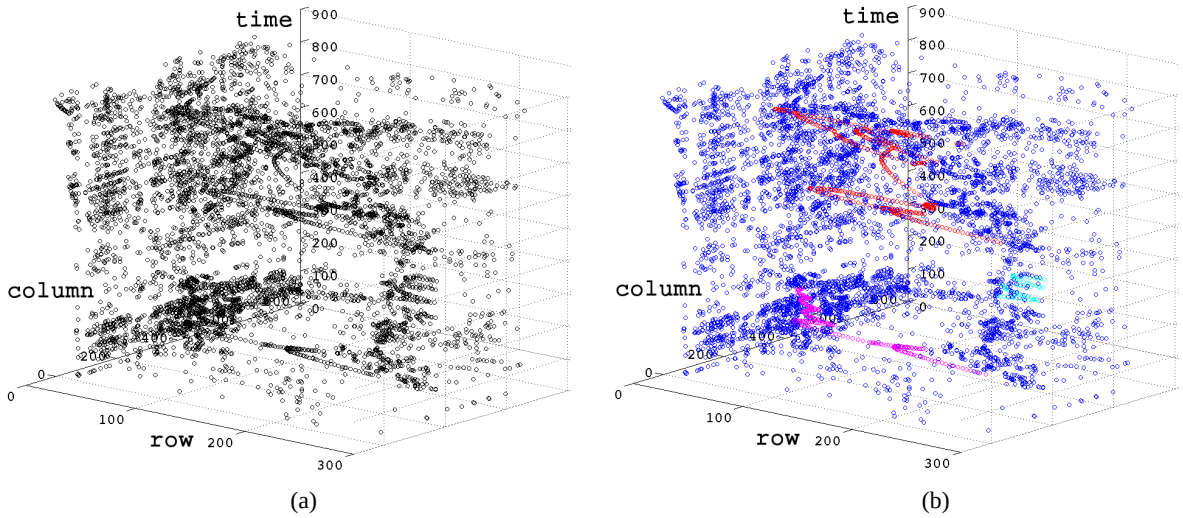


Fig. 2. Tennis ball tracking is equivalent to the task of recovering the color information in (b), assuming it is lost ((a)). (a): All ball candidates in a sequence plotted in a column-row-time 3D space. Each black circle is a ball candidate. This sequence contains 888 frames, and there are 6749 candidates generated. The average number of candidates is thus 7.6 per frame. The ball detection rate (the probability that a ball is detected as a candidate) is 90.7%. (b): Objective of tennis ball tracking: to recover the class label of each candidate: 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.

system using the homographies between the frames. Fig. 2 (b) shows the objective of tennis ball tracking: to recover the color information in this figure, assuming it is lost. In Fig. 2 (b), blue circles denote clutter-originated candidates, and circles with other colors 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 ball tracking is to identify the number of plays in this sequence, and identify all the ball-originated candidates in each play. As mentioned earlier, once this is achieved, the estimation problem, if still desired, is trivial. It should be noted that the second "play" 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 automatic tennis video annotation, it is left to a higher level reasoning tool to distinguish such a "play" from regular ones, when more information, such as player position and player action is available.

In this paper, we propose a three-layered data association scheme to achieve the task mentioned above. From bottom to top, the three layers are: candidate level association, tracklet level association, and path level analysis.

1) 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 optimized 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.
2) 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 of 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) 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 any pair of nodes in the graph, instead of the shortest path between a given pair of nodes. By analyzing 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.

These three layers of the proposed scheme will be described in more detail in the following three sections, respectively.

III. 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 optimized recursively to form a tracklet. After this process, the data association problem is transformed from the candidate space to a tracklet space.

A. Looking for a seed triplet

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, c_k^j is the j^{th} candidate in $\mathcal{C}_k$, and $k = 1, ..., K$. Assume a sliding window containing $2V+1$ frames is moving upward along the time axis in Fig. 2 (a). 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 $\mathcal{C}^{(i)} \triangleq \{\mathcal{C}_{i-V}, ..., \mathcal{C}_{i+V}\}$ onto the bottom plane in Fig. 2 (a), *i.e.* the row-column image plane. In this image plane, centered 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. 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 paper, we call the 3 candidates $c_{i-1}^{j'}$, c_i^j and $c_{i+1}^{j''}$ a **seed triplet**.

B. Fitting a model to the 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. After the projective transformation to the 2D image plane, the ball's motion is not strictly constant acceleration any more. However, according to perspective geometry [17], its motion is approximately constant acceleration, as long as ΔZ is negligible compared to Z, where ΔZ is the distance the ball moves in the world space between two frames along the direction that is perpendicular to the image plane, and Z is the distance from the ball to the camera along the same direction.

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 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:

$$\mathbf{v_1} = \frac{\mathbf{p_2}-\mathbf{p_1}}{\Delta k_{21}} - \frac{\Delta k_{21}\times\mathbf{a}}{2} \quad (1)$$

$$\mathbf{a} = 2 \times \frac{\Delta k_{21}\times(\mathbf{p_3}-\mathbf{p_2}) - \Delta k_{32}\times(\mathbf{p_2}-\mathbf{p_1})}{\Delta k_{21}\times\Delta k_{32}\times(\Delta k_{21}+\Delta k_{32})} \quad (2)$$

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 $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{a}, \mathbf{v_1} \in \mathbb{R}^2$. An estimate of the ball position in any frame k is then given by

$$\hat{\mathbf{p}}_\mathbf{k} = \mathbf{p_1} + \Delta k \times \mathbf{v_1} + \frac{(\Delta k)^2}{2} \times \mathbf{a} \quad (3)$$

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. 4 (a). In this special case, $k_1 = i - 1$, $k_2 = i$, and $k_3 = i + 1$.

C. Optimizing 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 [18]. 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. 4 (a). 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 optimizing the model using supports found in the previous iteration [19]. First, we define the support of a model to be 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 $d(\hat{\mathbf{p}}_\mathbf{k}, \mathbf{p_k^j}) < d_{th}$, where $d(\cdot,\cdot)$ is the Euclidean distance between two points; $\hat{\mathbf{p}}_\mathbf{k}$ is the estimated ball position at time k as given by the model; and d_{th} is a predefined threshold. The only exception here is that, in the rare case where more than one candidate satisfies this condition at time k, only the one with the smallest $d(\hat{\mathbf{p}}_\mathbf{k}, \mathbf{p_k^j})$ is selected as a support.

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

$$\begin{aligned} \dot{k} &\triangleq \min k \qquad \forall c_k^j \in \mathcal{S} \\ \dddot{k} &\triangleq \max k \qquad \forall c_k^j \in \mathcal{S} \\ \ddot{k} &\triangleq \arg\min_k ||\dddot{k} - k| - |k - \dot{k}|| \qquad \forall c_k^j \in \mathcal{S} \end{aligned} \quad (4)$$

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 optimization is thus complete (Fig. 4 (b)).

D. 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 [20], 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 [15], [21], which can be complicated and computationally expensive. In our implementation, the following cost function [20] is adopted:

$$C = \sum_{k=i-V}^{i+V} \sum_j \rho(\mathbf{p_k^j}) \quad (5)$$

where

$$\rho(\mathbf{p_k^j}) = \begin{cases} d^2(\hat{\mathbf{p}}_\mathbf{k}, \mathbf{p_k^j}) & \text{if } d(\hat{\mathbf{p}}_\mathbf{k}, \mathbf{p_k^j}) < d_{th} \\ d_{th}^2 & \text{if } d(\hat{\mathbf{p}}_\mathbf{k}, \mathbf{p_k^j}) \geq d_{th} \end{cases} \quad (6)$$

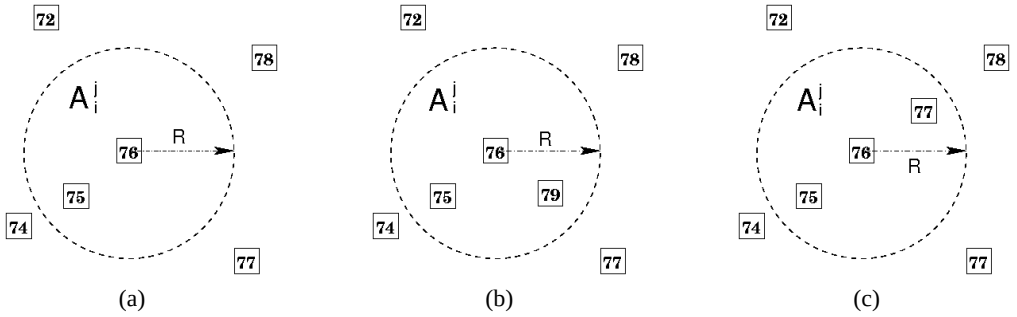


Fig. 3. Three examples of looking for seed triplet. Squares with numbers: candidates detected in different frames. Dashed circle: the circular area A_i^j around a candidate c_i^j. The radius of this circle is R. (a) and (b) For the candidate in frame 76, no seed triplet is found. Note that in (b), although 2 candidates (besides the one from frame 76) fall into the circular area, there is no candidate from $\mathcal{C}_{77}$. (c) Sufficient candidates are found in the circular area to form a seed triplet: one from $\mathcal{C}_{75}$ and one from $\mathcal{C}_{77}$.

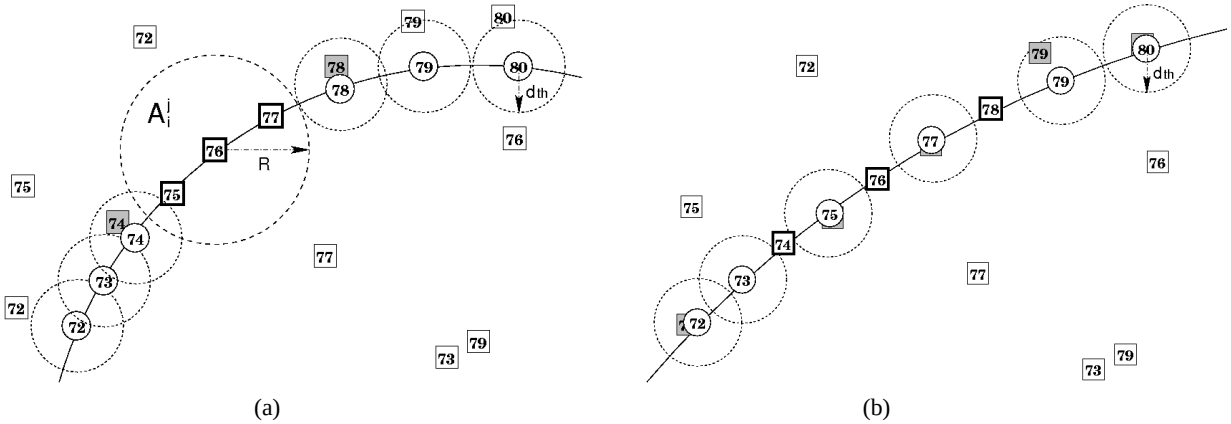


Fig. 4. Model fitting and model optimizing. Squares with numbers: candidates detected in different frames, including true positives and false positives. Big dashed circle: the circular area A_i^j around the candidate c_i^j. The radius of this circle is R. Bold squares: the triplet used for model fitting. 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. 3 (c). (b) Optimizing 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.* $\dot{k} = 74$, $\dddot{k} = 78$, and $\ddot{k} = 76$. The new triplet (bold squares in (b)) is then used to compute a "better" model.

and a smaller C indicates a better model.

Having defined C, the optimization loop terminates when the support set $\mathcal{S}$ stops expanding, or when C starts to increase. More specifically, let $M^{(z)}$ be the model after the z^{th} iteration, $C^{(z)}$ be its cost, $\dot{k}^{(z)}$ and $\dddot{k}^{(z)}$ are defined as in (4). The optimization loop terminates if

$$\dot{k}^{(z+1)} = \dot{k}^{(z)} \quad \wedge \quad \dddot{k}^{(z+1)} = \dddot{k}^{(z)} \tag{7}$$

or

$$C^{(z+1)} > C^{(z)} \tag{8}$$

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 observations becomes significant.

Once the optimization is complete, $M^{(z)}$ is retained as the final fitted model to the current seed triplet. If $M^{(z)}$ is ball-originated, it is now usually well converged to the "true motion" of the ball (see Fig. 5).

E. Definition of tracklets

Now we introduce the concept of **tracklet**. A tracklet T is defined as the combination of a parameterized 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 optimized motion model and its support set. For interval I_i, the above model-fitting/model-optimizing 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. 6 plots all 672 tracklets generated in an example sequence in the row-column-time 3D space. The candidates in this sequence were shown in Fig 2. In Fig. 6, 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 2 (a) and Fig. 6 (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

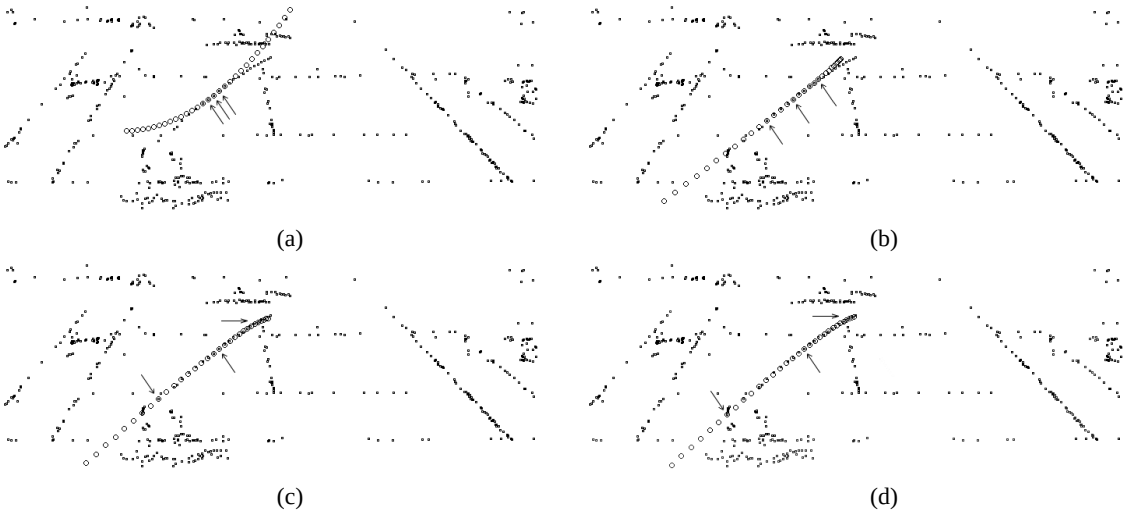


Fig. 5. Model fitting and model optimization on real data. Black squares: tennis ball candidates inside an interval of 31 frames ($V = 15$). The arrows point at the candidates that were used for model fitting. Circles: estimates given by the models. (a) Fitting a dynamic model to a seed triplet. (b) to (d): Three iterations of model optimization. Convergence is achieved after the 3rd iteration ((d)).

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).

IV. Tracklet Level Association

The generated tracklets may have originated from different balls or from clutter (see Fig. 6). 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 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.

A. 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

$$\mathcal{N} = \{\mathcal{N}_{1+V}, \mathcal{N}_{2+V}, ..., \mathcal{N}_{K-V-1}, \mathcal{N}_{K-V}\} \tag{9}$$

where the i^{th} stage $\mathcal{N}_i$ is the set of nodes (tracklets) generated in interval I_i, I_i centers 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

$$u < v \quad \wedge \quad \dot{k}_v^q - \dddot{k}_u^p \leq k_{th} \tag{10}$$

where $\dot{k}_v^q$ is the $\dot{k}$ of tracklet T_v^q, $\dddot{k}_u^p$ is the $\dddot{k}$ 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, where $k_{th} \in \mathcal{N}^+$. An example of graph topology is given in Fig. 7, 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

$$u < v \quad \wedge \quad \dot{k}_v^q - \dddot{k}_u^p \leq 0 \tag{11}$$

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

$$\begin{aligned} &(\exists c_k^{j'} \in \mathcal{S}_u^p \quad \wedge \quad \nexists c_k^{j''} \in \mathcal{S}_v^q) \quad \vee \\ &(\nexists c_k^{j'} \in \mathcal{S}_u^p \quad \wedge \quad \exists c_k^{j''} \in \mathcal{S}_v^q) \quad \vee \\ &(\exists c_k^{j'} \in \mathcal{S}_u^p \quad \wedge \quad \exists c_k^{j''} \in \mathcal{S}_v^q \quad \wedge \quad \mathbf{p}_{\mathbf{k}}^{\mathbf{j}'} \neq \mathbf{p}_{\mathbf{k}}^{\mathbf{j}''}) \end{aligned} \tag{12}$$

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

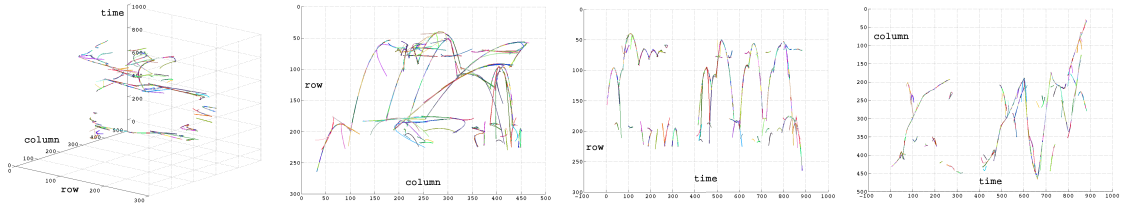
$$w_{u,v}^{p,q} \triangleq \begin{cases} \infty & \text{if } n_u^p \text{ and } n_v^q \text{ are conflicting} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

For non-overlapping nodes, their parameterized 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

$$w_{u,v}^{p,q} \triangleq \min d(\hat{\mathbf{p}}_{\mathbf{u},\mathbf{k}}^{\mathbf{p}}, \hat{\mathbf{p}}_{\mathbf{v},\mathbf{k}}^{\mathbf{q}}), \quad \forall k \in [\dddot{k}_u^p, \dot{k}_v^q] \tag{14}$$

B. 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.



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

Fig. 6. All the generated tracklets in the 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.

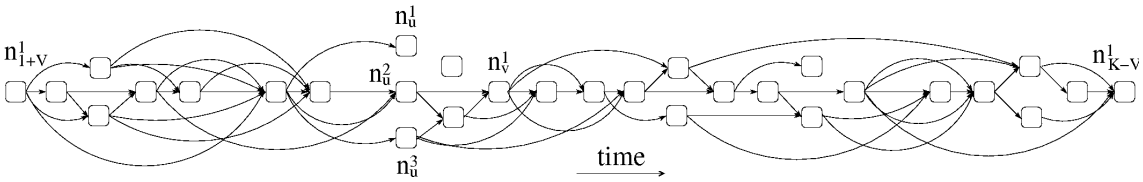
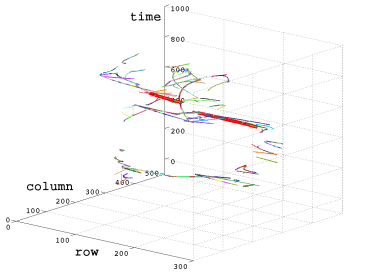
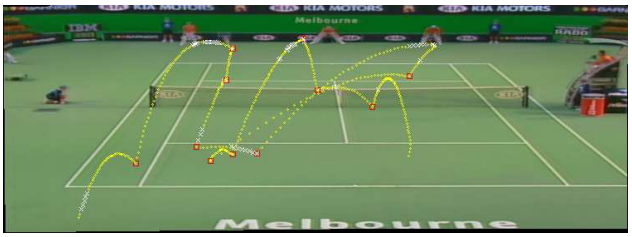


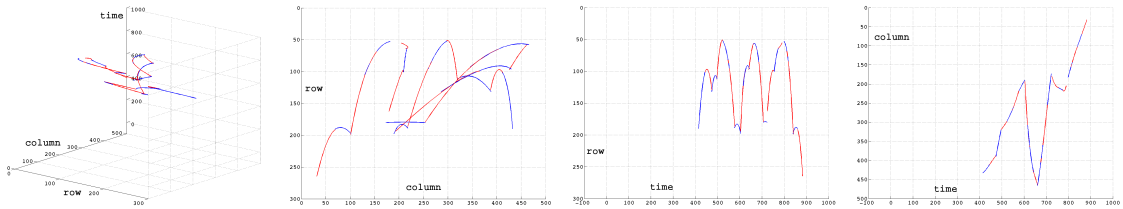
Fig. 7. An illustrative example of the graph topology. Nodes in the same stage are aligned vertically. Note that the number of nodes in a typical sequence is of the order of 10^2 to 10^3. Far fewer nodes are plotted in this figure for ease of visualization.



(a)



(f)



(b) (c) (d) (e)

Fig. 8. (a) 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. (b) to (e): Shortest path given by Dijkstra's algorithm. Adjacent nodes in the shortest path are plotted alternatively in blue and red. (b) 3D view. (c) Projection on the row-column plane. (d) Projection on the row-time plane. (e) Projection on the column-time plane. (f) 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.

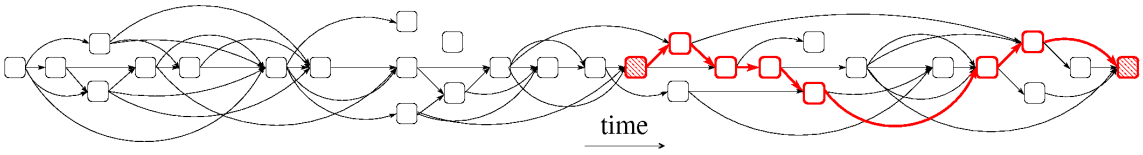


Fig. 9. An illustration of the shortest path in Fig. 8 in the graph. Striped red squares: the manually specified source node and destination node. Red squares: the shortest path between the source node and the destination node. The number of nodes in a typical sequence is of the order of 10^2 to 10^3. Far fewer nodes are plotted in this figure for ease of visualization.

The edge weight defined in (13) and (14) 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. In this section, 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 [22], 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 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. 8 (a)), respectively, and use Dijkstra's algorithm to find the shortest path between the two nodes.

Fig. 8 (b) to (e) show 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 (13) and (14), 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 generalized edge-preserving signal smoothing [23] to detect these key events. A more detailed description can be found in our previous work [24]. In Fig. 8 (f), the ball trajectory after interpolation and event detection is superimposed on the mosaic image.

V. 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. Note that the choice of APSP algorithm is not critical. 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.

A. 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 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 no matter how many balls there are to track, or where each of the ball trajectories starts and terminates in the graph, 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.

In order to achieve this, we need to define the relative quality and compatibility of the paths. First, 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 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". 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:

$$P_1 \left\{ \begin{array}{c} > \\ = \\ < \end{array} \right\} P_2 \quad \text{if} \quad (W_1 - W_2) \left\{ \begin{array}{c} < \\ = \\ > \end{array} \right\} \alpha \cdot (L_1 - L_2) \quad (15)$$

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 [25], $\mathcal{Q}$ 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. It should also be noted that the definition of path compatibility is different from that of tracklet compatibility in Section IV-A.

- **Initialisation:** $\mathcal{P}$ has p paths, and $\mathcal{B}$ is empty.
- **While** $\mathcal{P}$ is not empty:
  - Remove the best path P^* in $\mathcal{P}$ from $\mathcal{P}$;
  - If P^* is compatible with all paths in $\mathcal{B}$, add P^* to $\mathcal{B}$.

Algorithm 1: the PR algorithm

Now we propose a simple Paths Reduction (PR) algorithm (see Algorithm 1) to reduce the size of 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 PR algorithm the Best Set of Compatible Paths (BSCP), and denote it by $\mathcal{B}$.

B. Applying the PR algorithm to $\mathcal{Q}$

Now we finally have a complete data association algorithm for tracking multiple objects fully automatically. We apply the complete algorithm to the example sequence. In Section IV-B, 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. 10. In Fig. 11, the 3 reconstructed trajectories are superimposed on mosaic images. An illustration of the 3 paths in the graph is shown in Fig. 12.

VI. 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.

A. 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 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.

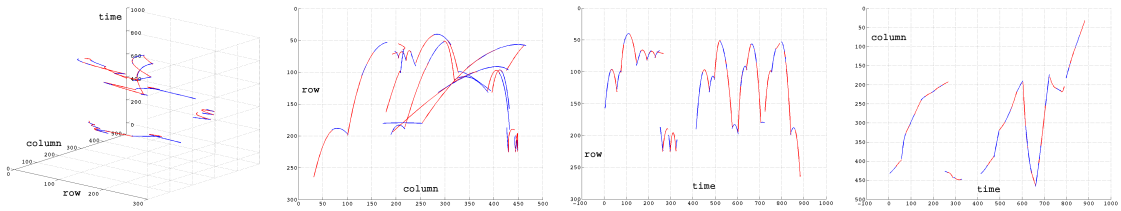
B. Tracklet level association

At the tracklet level, we need to solve an APSP problem for a graph $\mathcal{G}$ with N nodes. N is normally of the order of $10^2 \sim 10^3$. Obviously, Dijkstra's algorithm can be applied repeatedly to solve an APSP problem. However, the efficiency of such a scheme will be low. Several dedicated APSP algorithms have been proposed. The Floyd-Warshall algorithm solves APSP on a directed graph in $O(N^3)$ time [26]. 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 [27].

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. 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 IV-A is not a trellis. However, it satisfies a weaker condition. According to (9) and (10), $\mathcal{N}$ 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 from nodes in stage $\mathcal{N}_u$ to nodes in stage $\mathcal{N}_v$ only 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.

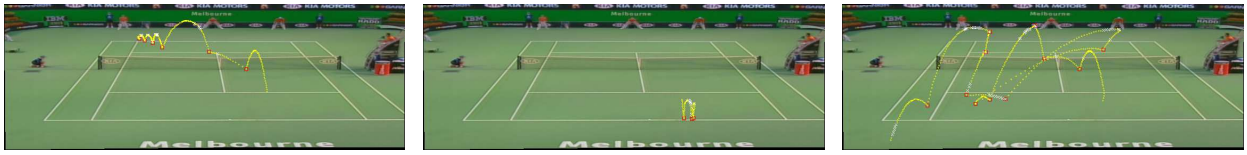
The proposed APSP algorithm uses the concept of dynamic programming. Suppose we are in the middle of the tracklet generation process. The sliding window now centers on frame $i-1$, and tracklets in interval I_{i-1} have been generated. Let $\mathcal{G}^{(i-1)} = \{\mathcal{N}^{(i-1)}, \mathcal{E}^{(i-1)}\}$ 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. That is, 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. With these two pieces of information for each node $n_u^p \in \mathcal{G}^{(v-1)}$ in each node $n_v^q \in \mathcal{G}^{(i-1)}$, the shortest path between any pair of nodes in $\mathcal{G}^{(i-1)}$ can be identified by back tracing.

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 centers on frame i. Assume several tracklets are generated in



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

Fig. 10. 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.



(a) (b) (c)

Fig. 11. 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.

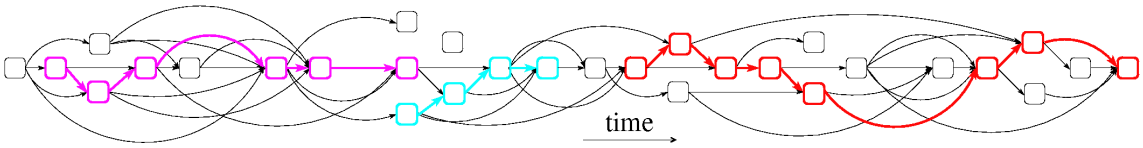


Fig. 12. An illustration of the paths in $\mathcal{B}_{th}$. Magenta, cyan, and red paths correspond to the first, second and third play in the sequence, respectively.

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. 13, the sub-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 $n_{u_1}^{p_1}, n_{u_2}^{p_2}, ..., n_{u_s}^{p_s}$, and are plotted as shaded nodes in Fig. 13. Edges that connect these nodes to n_i^l are denoted by $e_{u_1,i}^{p_1,l}, e_{u_2,i}^{p_2,l}, ..., e_{u_s,i}^{p_s,l}$, 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. 13, 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 $n_{u_1}^{p_1}, n_{u_2}^{p_2}, ..., n_{u_s}^{p_s}$ and the corresponding edge in $e_{u_1,i}^{p_1,l}, e_{u_2,i}^{p_2,l}, ..., e_{u_s,i}^{p_s,l}$. 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 $n_{u_r}^{p_r}$ is kept in the table in $n_{u_r}^{p_r}$, where $r = 1, 2, ..., s$. Let $W^{(i-1)}(n_u^p, n_{u_r}^{p_r})$ be the total weight of the shortest path in $\mathcal{G}^{(i-1)}$ from n_u^p to $n_{u_r}^{p_r}$, as kept in the table in node $n_{u_r}^{p_r}$. In the special case where the table in $n_{u_r}^{p_r}$ does not contain an entry for n_u^p, *i.e.* $u_r \leq u$, we define $W^{(i-1)}(n_u^p, n_{u_r}^{p_r}) = \infty$. The total weight of the shortest path in $\mathcal{G}^{(i)}$ from n_u^p to n_i^l is then:

$$W^{(i)}(n_u^p, n_i^l) = \min[W^{(i-1)}(n_u^p, n_{u_r}^{p_r}) + w_{u_r,i}^{p_r,l}], \forall r \in [1, s] \tag{16}$$

where $w_{u_r,i}^{p_r,l}$ is the weight of edge $e_{u_r,i}^{p_r,l}$. 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

$$\{u^*, p^*\} = \arg \min_{\{u_r, p_r\}} [W^{(i-1)}(n_u^p, n_{u_r}^{p_r}) + w_{u_r,i}^{p_r,l}], \forall r \in [1, s] \tag{17}$$

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 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 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. The proposed APSP algorithm is summarised as Algorithm 2.

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

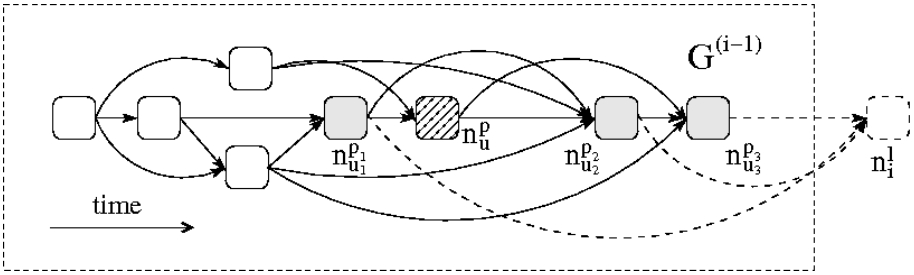


Fig. 13. Constructing the table of APSP knowledge for a node $n_i^l \in \mathcal{N}_i$. The sub-graph in the big rectangle: $\mathcal{G}^{(i-1)}$. Dashed node: a node $n_i^l \in \mathcal{N}_i$. Shaded nodes: nodes in $\mathcal{G}^{(i-1)}$ that are connected to n_i^l with edges. Striped node: an arbitrary node $n_u^p \in \mathcal{G}^{(i-1)}$.

- **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 an entry labelled n_u^p to the table of APSP knowledge in n_i^l;
    - put $W^{(i)}(n_u^p, n_i^l)$ and $n_{u*}^{p^*}$ given by (16) and (17) into this entry.

Algorithm2: the proposed APSP algorithm.

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 $\mathcal{G}$.

C. 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 (15). 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 $10^4 \sim 10^6$. The paths in $\mathcal{Q}$ are stored in a linked list, and a merge sort implementation for linked list with improved memory efficiency [28] 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.

The speed of each layer of the proposed algorithm on 5 sequences of various lengths is shown in Table I. The longest sequence among them contains 1266 frames. It is also the longest sequence in the three data sets used in the experiments (see next section for details of the data sets). As discussed earlier in this section, some components of the proposed algorithm have linear complexity in the length of the sequence, while others have super-linear complexity. As a result, the running time of the complete algorithm increases faster than the length of the sequence. This is evident in Table I: as both the sequence length and running time increase, the ratio of them decreases. In sequence 5, the ratio even drops below 1. However, even on sequence 5, the running time is comparable to the length of the sequence. It should be noted that since the proposed data association algorithm takes ball candidate positions as input, the time spent on image processing, *i.e.* extracting ball candidates, is not included in the total running time in Table I. Due to the use of homographies for motion compensation between frames, the image processing part is quite expensive. This could be improved by approximating the homographies using more efficient methods, however, we have focused on the data association problem, and so far have not addressed the efficiency of image processing.

VII. Experiments

A. 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 (see Fig. 14 for examples), and the camera has small pan, tilt and zoom (PTZ) motion. Tennis ball tracking is performed only on play shots. In this paper, we also refer to a play shot as a "sequence".

We used three data sets from three different tournaments containing 165 sequences in total. Example frames from the three data sets can be found in Fig. 14. One data set with 25 sequences was used as training set, and the other two with 70 sequences each were used as test sets. Some statistics of the three data sets are shown in Table. II. In Table. II, $\bar{N}$ 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 algorithm were tuned using the training set. The parameters were assumed independent of each other, and were tuned separately to maximise the F-measure of event detection. The definition of the F-measure of event detection will be given shortly in Section VII-E. 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 $\alpha = 1.0$. Once the optimal set of parameter values was obtained, the proposed algorithm was then applied to the test sets.

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 other papers, we have used, and will continue using the word "frame" to refer to "field". As a result, the "frame" rate of the sequences is 50 per second. After de-interlacing, homographies between frames were calculated by tracking corresponding corners, and then used to compensate the global motion (camera PTZ) between frames. Foreground moving objects were then extracted by

TABLE I
RUNNING TIME OF EACH COMPONENT OF THE PROPOSED ALGORITHM ON 5 SEQUENCES

sequence id.		1	2	3	4	5
statistics of sequences	number of frames	138	396	610	888	1266
	length of sequences (s)	**2.76**	**7.92**	**12.20**	**17.76**	**25.32**
	number of nodes in $\mathcal{G}$	101	316	490	672	1190
	number of paths in $\mathcal{Q}$	757	17635	47723	87961	328790
running time (s)	image processing	18.22	52.18	81.20	117.35	169.03
	tracklet generation	0.67	1.42	1.65	1.91	2.80
	APSP	0.59	2.61	5.72	12.58	24.03
	ordering the paths	0.0003	0.005	0.009	0.03	0.23
	path reduction (PR)	0.02	0.15	0.41	0.87	5.74
	total exclud. image processing	**1.28**	**4.19**	**7.79**	**15.40**	**32.80**
ratio of seq. length in seconds to total running time		**2.16**	**1.89**	**1.57**	**1.15**	**0.77**

TABLE II
STATISTICS OF THE EXPERIMENTAL DATA

	game	court type	# of seq.	# of frames	length	$\bar{N}$	r_d
training set	Australian'03 Women's Final	artificial	25	15716	5.2 min	12.1	90.3%
test set 1	Australian'06 Men's Final	artificial	70	60094	20.0 min	10.6	91.6%
test set 2	Wimbledon'05 Men's Semi Final	grass	70	43706	14.6 min	16.5	88.2%

(a) (b) (c)

Fig. 14. Three example frames from the three data sets, one frame from each set. (a) training set. (b) test set 1. (c) test set 2. The three sets have different court types and have different illumination conditions. Note the illumination difference between (a) and (b).

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 the proposed algorithm. Note that only weak prior knowledge (size) was used for deciding whether a foreground blob was a candidate. This inevitably introduced more false candidates. The idea was to test the proposed algorithm under heavily cluttered environments. Moreover, since only a weak prior was used, when the proposed algorithm is applied to sequences with different court type, or recorded with a different illumination conditions (see Fig. 14 for examples), little retraining is required. We will see this shortly in the experimental results.

B. Algorithms for comparison

For comparison with the proposed algorithm, PDA [2] and RDA [15] were also implemented. Due to the abrupt motion changes, PDA 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. Since PDA failed completely, its performance is not shown here. We will 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 [29] that

$$n \geq \frac{\log(1-P_0)}{\log(1-\epsilon^s)} \tag{18}$$

where ϵ 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 [15]. According to (18), at least 8161 trials were needed. The mixture parameters of the likelihood function in RDA were estimated recursively until convergence, also as suggested in [15]. The dynamic model fitted to a sample set of three candidates was the same as used in the proposed algorithm: a constant acceleration model as defined in (1) and (2). Since RDA by itself cannot deal with track initiation and track termination, in our experiments, for each play in a sequence, we manually specified the frames in which RDA was applied.

C. Marking up the ground truth

For each sequence, we manually marked up the ball positions in each frame, and different balls were assigned different labels. If a ball was invisible for no more than k_{th} frames, its positions in the "invisible frames" were marked up by manual interpolation. Note that the threshold k_{th} is the same as the one used in (10). Table III demonstrates how to decide when "manual interpolation" is applied using a synthesized example. In the short sequence in Table III, there are two balls,

TABLE III
DECIDING WHEN TO INTERPOLATE BALL POSITIONS IN GROUND TRUTH

ball \ frame	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1		o	o	o	-	o	o					o	o	o	o	o	
2								o	o	o	-	-	o	o	o		

labeled as ball 1 and ball 2. A grid with an "o" means the corresponding ball in the corresponding frame is visible, and its position is marked up. For any other grid, the corresponding ball in the corresponding frame is not visible. In grids with "-", ball positions are interpolated, because the duration of the "invisible frames" is no more than k_{th}. In this example, k_{th} is set to 3. After manual interpolation, the ground truth is complete: 3 ball trajectories that have originated from 2 balls. The 3 trajectories span frame 2 to 7, frame 8 to 15, frame 12 to 16, respectively. For each trajectory, ball position in each frame is marked up.

D. 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. 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. 15. Since in practice $\mathcal{T}_{gt}$ and $\mathcal{T}_{tr}$ contain small numbers of trajectories, typically between 1 and 5, 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 correspondence pairs, where each pair is a correspondence between one trajectory in $\mathcal{T}_{gt}$ and one in $\mathcal{T}_{tr}$. For a given pair, let $\mathcal{I}^*$ be the time interval where the two trajectories overlap in time. For each frame in $\mathcal{I}^*$, 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. A correspondence between the trajectories in $\mathcal{T}_{gt}$ and the ones in $\mathcal{T}_{tr}$ is thus obtained automatically.

E. Evaluation methods

a) 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. We first define the quality of track initiation and track termination. The best assignment scheme consists of several correspondence pairs. Let $T^i_{gt} \in \mathcal{T}_{gt}$ and $T^i_{tr} \in \mathcal{T}_{tr}$ be the trajectories in the i^{th} pair, and $\mathcal{I}^i_{gt}$ and $\mathcal{I}^i_{tr}$ be the time interval of T^i_{gt} and T^i_{tr}, respectively. In the general case, there may be one or more trajectories in $\mathcal{T}_{gt}$ and/or in $\mathcal{T}_{tr}$ that are not in any correspondence pair. Let $T^{'j}_{gt}$ be the j^{th} trajectory in $\mathcal{T}_{gt}$ that is not in any correspondence pair, and $\mathcal{I}^{'j}_{gt}$ be its time interval. Also let $T^{'k}_{tr}$ be the k^{th} trajectory in $\mathcal{T}_{tr}$ that is not in any correspondence pair, and $\mathcal{I}^{'k}_{tr}$ be its time interval. Now define

$$I^{\cap} \triangleq \sum_i |\mathcal{I}^i_{gt} \cap \mathcal{I}^i_{tr}| \tag{19}$$

and

$$I^{\cup} \triangleq \sum_i |\mathcal{I}^i_{gt} \cup \mathcal{I}^i_{tr}| + \sum_j |\mathcal{I}^{'j}_{gt}| + \sum_k |\mathcal{I}^{'k}_{tr}| \tag{20}$$

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

$$\beta \triangleq \frac{I^{\cap}}{I^{\cup}} \tag{21}$$

where β ranges between 0 an 1. When $\beta = 1$, track initiation/termination is said to be perfect. An illustration of how β is defined is shown in Fig. 16.

The quality of track initiation and termination over several sequences is defined as

$$\beta \triangleq \frac{\sum_m I^{\cap}_m}{\sum_m I^{\cup}_m} \tag{22}$$

where $I^{\cap}_m$ and $I^{\cup}_m$ are the $I^{\cap}$ and $I^{\cup}$ of the m^{th} sequence, respectively.

b) 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 $\mathcal{I}^i_{tr} \cap \mathcal{I}^i_{gt}$, one by the tracker and the other by the ground truth. We define the tracking error to be the Euclidean distance between these two positions. 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. Once tracking error is defined, an LOT is then defined as when the tracking error is greater than 6 pixels. Let the number of LOTs in the m^{th} sequence be I^L_m. The proportion of LOT on several sequences is then defined as:

$$\gamma \triangleq \frac{\sum_m I^L_m}{\sum_m I^{\cap}_m} \tag{23}$$

c) Quality of event detection: We also measure the performance of the proposed algorithm in terms of quality of event detection. In a tennis game, an event can be a ball bouncing on the ground, a player hitting the ball, or a ball hitting the net. In other words, an event is a motion discontinuity in the

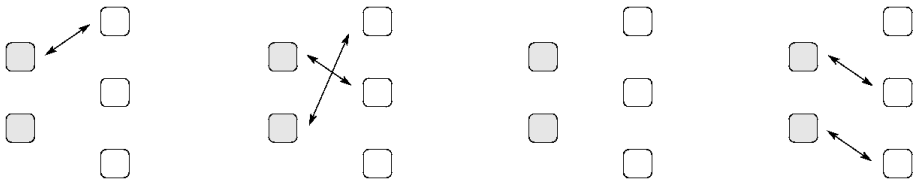


Fig. 15. Four examples of valid assignment schemes. Shaded squares: trajectories in $\mathcal{T}_{gt}$. White squares: trajectories in $\mathcal{T}_{tr}$.

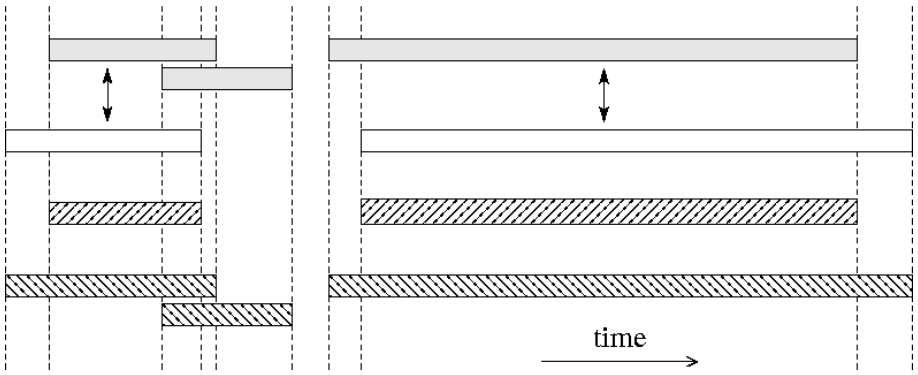


Fig. 16. An illustration of the definition of quality of track initiation and track termination. Each bar in the figure represents a time interval. The length of each bar is proportional to the length of the time interval. Shaded bars: time intervals of the trajectories in $\mathcal{T}_{gt}$. White bars: 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 $I^{\cup}$.

trajectory. Recall that each trajectory is a sequence of points in the row-column-time space. For each trajectory in $\mathcal{T}_{gt}$, points that correspond to motion discontinuities are marked up as ground truth events. 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^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 recall and precision of event detection are then defined as:

$$\text{Precision} = \frac{\text{number of matched events}}{\text{number of detected events}} \quad (24)$$

$$\text{Recall} = \frac{\text{number of matched events}}{\text{number of events in the ground truth}} \quad (25)$$

We can then calculate the F-measure of event detection, which is defined as the harmonic mean of the precision and the recall:

$$F \triangleq \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (26)$$

where F 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 being 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.

d) Processing speed: In Table I in the previous section, we showed the speed of each component of the proposed algorithm on 5 example sequences. In this section, we present in Table. IV the overall processing speed of the proposed algorithm and LDA in terms of "frames per second". Note that, as in Table I, when calculating the speed in Table. IV, the time spent on image processing is not included.

TABLE IV
EXPERIMENTAL RESULTS ON THE TEST SETS.

		β	γ	F-Measure	frames/s
test set 1	proposed	83.7%	3.51%	0.758	98.4
	RDA	–	11.74%	0.423	3.6
test set 2	proposed	82.2%	3.66%	0.729	109.3
	RDA	–	13.23%	0.411	3.2

F. Experimental results and discussion

Experimental results on the two test sets are shown in Table IV. The quality of track initiation/termination of the two algorithms is not comparable, since in RDA this was manually dealt with. We now discuss the experimental results with respect to the other three measurement metrics. First, on both test sets, the proportion of LOT of the proposed algorithm is much lower than that of RDA. Very briefly speaking, this is because RDA operates only at a low level. In RDA, estimates are given by the best models in corresponding intervals independently of each other, and the concept of "tracklet" does not exist. This significantly limits RDA's ability to deal with complexities in broadcast tennis video, such as the presence of multiple balls, non-uniform non-white background clutter, sudden change of tennis ball motion, etc. 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 three 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. As a result, a model fitted to a ball that is not being tracked can be "better" than that fitted to the ball being tracked.
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 candidates that have all originated from the ball being tracked can have a poor quality, if there is an event in between the three candidates.

In any of the above three cases, the assumption for RDA to work breaks down.

This can be observed in Fig. 17. Fig. 17 (b) shows the tracking results of RDA on a sequence with only one play. We can see from Fig. 17 (b) that in some sliding window positions, models fitted to candidates that have originated from clutter (in this case from the near player) are selected as the best models. Since in RDA estimates are given by the best models in corresponding intervals independently of each other, no motion smoothness constraint is applied. Such errors cannot be recovered. We can also see from Fig. 17 (b) that the reconstructed trajectory given by RDA tends to be smooth when the ball change its motion abruptly, for example, when the near player serves. This is because RDA does not handle sudden motion change of the ball explicitly. It always tries to fit a constant acceleration model to three candidates, disregarding the fact that candidates near a key event can only be explained by two constant acceleration models.

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. For example, candidate level association focuses on growing candidates into tracklets. Tracklet level association focuses on linking tracklets into paths, where by carefully defining edge weight, the sudden change of ball motion is also tackled. Finally, the path reduction algorithm solves the association problem at the path level. By appropriately slicing the problem into layers, the proposed method is much more flexible than RDA in dealing with the complexities in broadcast tennis video.

The quality of event detection in terms of F-Measure of both algorithms are also compared in Table IV. The quality of event detection is affected directly by the quality of the tracking. This partly explains why the proposed method has a higher F-measure than RDA. As mentioned earlier, the reconstructed trajectory given by RDA tends to be smooth when the ball changes its motion abruptly. Since events are detected by identifying motion discontinuities, this also degrades the performance of RDA in terms of event detection. For example, the first event in this sequence, the serve of the near player, is not detected, resulting in a false negative. Also, the detected position of the second event, the ball bouncing at far side of the court, is not exactly where it should be, due to this smoothing effect.

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 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 optimization to converge. The effective number of models evaluated is then $10.6 \times 1.02 \times 3.7 \approx 40$. The various speeding up techniques discussed in Section VI, especially the proposed APSP algorithm, also help make the proposed method more efficient. On the other hand, according to Eq. (18), 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 I and Table IV was measured on a single thread of an Intel Xeon 3.0G computer running Linux, and overhead of disk I/O was included. Both the proposed algorithm and RDA were implemented in C++.

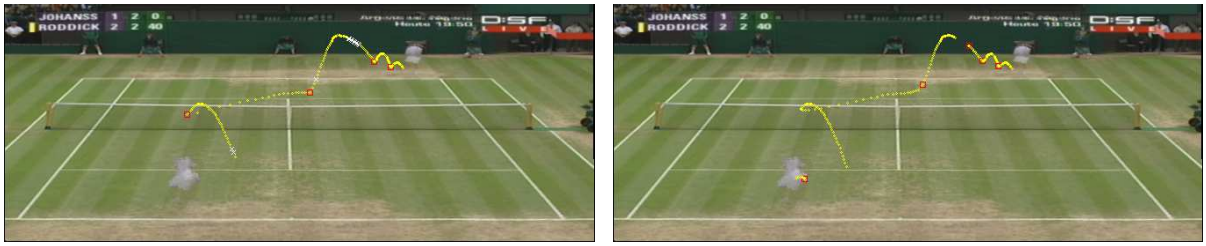
G. Sensitivity to parameters

In Table V we show the sensitivity of the proposed algorithm to parameters using test set 1. The optimal set of parameters obtained in the training phase was used as a baseline configuration. In each experiment, we increased or decreased each of the parameters, and measured the performance of the proposed algorithm using the modified set of parameters.

We can see from the tables that the performance of the proposed algorithm in terms of event detection is not sensitive to small changes of most parameters. However, when the parameter for determining the relative quality of two paths, α, varies over a large range, the performance is affected. When an inappropriate value for α is used, at the path level, paths that corresponds to multiple plays can be merged into one path, or a path that corresponds to one play can be broken into multiple paths (see Fig. 18 for an example). According to the way the tracked trajectories and ground truth trajectories are matched, this can severely degrade the performance of track initiation/termination, and in turn degrade the performance of 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, many ball-originated tracklets are filtered out. The shortest path algorithm is forced to take a wrong path. On the other hand, when m_{th} is low, many false tracklets are allowed. However, the shortest path algorithm can remove them effectively.

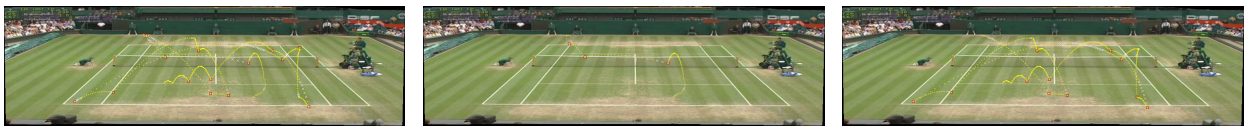
VIII. Conclusions

In this paper, we have presented a multi-layered data association algorithm with graph-theoretic formulation for tracking multiple objects that undergo switching dynamics in clutter. There is a natural connection between graphs and data association. Previous work can be found in the literature on using graph theory to solve the data association problem, *e.g.* [3], [30]. Usually each object candidate is modeled as a node in a graph, and the object trajectory is identified by looking for the "optimal" path in some sense. Typically, the graph constructed is a trellis, and the algorithm used for searching the optimal path is the Viterbi algorithm. The main difference between our approach and the previous work is that, before constructing a graph, we map the data association problem



(a) (b)

Fig. 17. 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.



(a) (b) (c)

Fig. 18. (a) A trajectory given by the optimal set of parameters. (b) and (c): This trajectory is broken into two if $\alpha = 0.01$.

TABLE V
SENSITIVITY TO PARAMETERS.

		β	γ	F-Measure	frames/s
opt. param.		83.7%	3.51%	0.758	98.4
RDA		-	11.74%	0.423	3.6
V	7	82.8%	3.94%	0.725	102.2
	13	82.9%	3.78%	0.770	93.8
R	15.0	83.6%	3.58%	0.754	100.3
	25.0	83.7%	3.50%	0.759	96.6
d_{th}	4.0	82.9%	3.87%	0.732	128.4
	6.0	83.0%	3.29%	0.768	81.4
m_{th}	4	83.0%	3.82%	0.763	55.7
	8	75.9%	3.75%	0.696	152.9
k_{th}	15	76.5%	2.78%	0.740	105.6
	25	85.1%	4.46%	0.747	89.4
L_{th}	30	80.0%	2.89%	0.747	92.9
	60	84.0%	3.29%	0.759	102.4
α	0.001	42.5%	2.10%	0.538	88.8
	0.01	49.3%	2.14%	0.584	91.3
	0.1	78.1%	2.64%	0.738	96.0
	10.0	76.8%	5.78%	0.700	98.5
	100.0	75.2%	7.48%	0.668	100.0
	1000.0	75.1%	7.60%	0.667	100.2

from a candidate space to a tracklet space, where it is easier to deal with. We then map the association problem onto a graph, and solve it using 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. By exploiting a special topological property of the graph, we have also developed an all-pairs shortest path algorithm which is more efficient than the general-purpose ones. 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.

ACKNOWLEDGEMENTS

This work has been supported by EU IST-2001-34401 VAMPIRE Project, EU IST-507752 MUSCLE Network of Excellence, and EU IST-FP6-045547 VIDI-Video Project.

REFERENCES

[1] P. Smith and G. Buechler, "A branching algorithm for discriminating and tracking multiple objects," *IEEE Transactions on Automatic Control*, pp. 101–104, 1975.
[2] Y. Bar-Shalom and T. E. Fortmanm, *Tracking and Data Association*. Academic Press INC., 1988.
[3] T. Quach and M. Farooq, "Maximum likelihood track formation with the viterbi algorithm," in *IEEE Conference on Dedsion and Control*, 1994, pp. 271–276.
[4] R. Streit and T. Luginbuhl, "Probabilistic multi-hypothesis tracking," Technical Report, 1995.
[5] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24(6), pp. 843–854, 1979.
[6] Y. Bar-shalom, X. Li, , and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, 2001.
[7] P. Willett, Y. Ruan, and R. Streit, "The pmht for maneuvering targets," in *SPIE Conference on Signal and Data Processing of Small Targets*, 1998, pp. 416–427.
[8] 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*, vol. 34(1), pp. 103–122, 1998.
[9] D. Musicki, B. F. L. Scala, and R. J. Evans, "Integrated track splitting filter for manoeuvring targets," in *IEEE International Conference on Information Fusion*, 2004.
[10] S. S. Blackman, M. T. Busch, and R. F. Popoli, "Imm-mht tracking and data association for benchmark tracking problem," in *American Control Conference*, vol. 4, 1995, pp. 2606–2610.
[11] N. Owens, C. Harris, and C. Stennett, "Hawk-eye tennis system," in *International Conference on Visual Information Engineering*, 2003.
[12] 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*, 1998, pp. 260–265.
[13] 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*, vol. 2, 2004, pp. 1049–1052.
[14] 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*, 2000, pp. 320–325.
[15] V. Lepetit, A. Shahrokni, and P. Fua, "Robust data association for online applications," in *IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, 2003, pp. 281–288.

[16] 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*, vol. 24(6), pp. 381–395, 1981.
[17] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.
[18] 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*, 2002, pp. 458–467.
[19] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *DAGM-Symposium*, 2003, pp. 236–243.
[20] P. H. S. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.
[21] 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*, vol. 27(10), pp. 1523–1535, 2005.
[22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
[23] V. Mottl, A. Kostin, and I. Muchnik, "Generalized edge-preserving smoothing for signal analysis," in *IEEE Workshop on Nonlinear Signal and Image Analysis*, 1997.
[24] 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.
[25] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*. Cambridge University Press, 2002.
[26] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5(6), p. 345, 1962.
[27] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *Journal of the ACM*, vol. 24(1), pp. 1–13, 1977.
[28] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press, 1990.
[29] O. Chum, J. Matas, and S. Obdrzalek, "Enhancing ransac by generalized model optimization," in *Asian Conference on Computer Vision*, vol. 2, 2004, pp. 812–817.
[30] 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*, vol. 25, pp. 287–296, 1989.