A TWO LAYERED DATA ASSOCIATION APPROACH FOR BALL TRACKING

Xiangzeng Zhou¹, Qiang Huang², Lei Xie¹, Stephen J. Cox²

¹School of Computer Science, Northwestern Polytechnical University, Xi’an, 710072, P.R. China
²School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, United Kingdom

ABSTRACT
Ball-tracking is a key technology in processing and analyzing a ball game. Because of the complexity of visual scenes, a large number of objects are usually selected as candidates for the ball, leading to incorrect identification, and conversely, the true position of the ball may sometimes be missed. In this paper, we propose a two layered data association method to improve the robustness of ball-tracking. At a local layer, we use a sliding window based Token Transfer method to generate a set of sub-trajectory candidates. At a global layer, a single ball trajectory is obtained by applying a dynamic programming based splice method to a graph consisting of the sub-trajectory candidates. We evaluated our approach on tennis matches from the Australian Open and the U.S. Open, and the results obtained show that our approach outperforms the state-of-art approach by around 30 %.

Index Terms— data association, tennis, ball tracking, trajectory, layered

1. INTRODUCTION
Sports video analysis is currently receiving increasing attention. It has a number of useful and beneficial applications, such as highlight extraction [1], tactics analysis [2], computer-assisted refereeing [3], etc. Robust detection and tracking of figures and objects in the game is the foundation for high level analysis. Knowledge of the position of the ball at any time is also essential for more ambitious systems that attempt to “understand” a game [4].

This paper is concerned with ball tracking, which has been traditionally approached as a data association task using a Markov chain assumption and a predictive motion model, applied frame by frame. However, data association in candidate “clutter” is difficult because of false positives (non-ball objects detected as balls) and false negatives (undetected balls). After probabilistic data association (PDA) was first proposed by Bar-Shalom and Fortmann [5], many researchers have strived to improve the robustness of this task [6] [5] [7]. Viterbi Data Association (VDA) [7] uses a parallel search scheme using the Viterbi algorithm. Robust data association (RDA, [6]) treats data association as a motion fitting problem in an attempt to provide robustness to abrupt motion change (e.g. when the ball is struck). Yan et al.’s work [8] describes a hierarchical scheme with a graph-theoretic formulation that attempts to overcome some of RDA’s limitations (e.g. the lack of motion smoothness constraints). More recently, Huang et al. [9] described a Viterbi-based method to estimate ball trajectory: however, it loses some precision in tracking the target, mainly because of the lack of a well-defined motion model.

In this paper, we describe a two layered approach for tennis ball tracking that attempts to increase robustness against both abrupt motion change and the absence of the tracked object. Given a set of ball candidates, at the “local” layer we use a Token Transfer method to generate a set of sub-trajectories as the input to the next layer. At the “global” layer, we use a dynamic programming based splice scheme to obtain the optimal sub-trajectory combination as the final trajectory. The paper is organized as follow. Ball candidate extraction and the local method are described in Section 2. A description of the global layer approach is given in Section 3. Evaluation results are shown in Section 4, and conclusions are presented in Section 5.

2. LOCAL LAYER: N-BEST SUB-TRAJECTORIES EXTRACTION

2.1. Candidates Extraction
To extract ball candidates in each frame, we difference two adjacent frames and choose candidates using size and color filters. Two kinds of mask are then used to discard false candidates. Firstly, we extract the court lines using the direct linear transformation (DLT, [10]) to discard those candidates that occur on the court. Secondly, we apply a mean shift based method [11] to track players and discard ball candidates that occur inside players’ regions. We denote the set of ball candidates extracted in frame $f_i$ as $C_i = \{C_1^i, C_2^i, \ldots, C_N^i\}$ ($i = 1 \ldots T$).
Fig. 1. Prediction of ball position using a linear acceleration model and detected ball candidates. Filled circles represent actual detected ball candidate positions and the dotted circle a predicted position.

2.2. Motion Model and Quality Score

Next, we define a directed network using the following link rule: each candidate can link forwards to any candidate located in its neighborhood \([mTh, MTh]\) in the next frame. \(MTh\) and \(mTh\) represent the maximum and minimum pixel distances a tennis ball can travel in \(\Delta T\) (the reciprocal of the frame rate); these values are here set experimentally to 65 and 5 respectively. The nodes of the resulting network are termed Candidate Nodes (CNs), and each path through the network corresponds to a sub-trajectory candidate.

To predict the position of a ball in frame \(f_{t+1}\), we “score” each candidate \(C_{t+1}^n\) using a local motion model estimated using the positions of previous candidates in three previous frames, and the predicted candidate \(C_{t+1}^*\) given by the motion model. The local motion model assumes constant acceleration. Velocity \(V_t^k\) and acceleration \(Acc_t^k\) are estimated using equations 1 and 2 below, and are used to predict a candidate position \(\hat{C}_{t+1}^*\) in equation (3).

\[
\begin{align*}
Acc_t^k &= \frac{(C_t^k - C_{t-1}^i) - (C_{t-1}^i - C_{t-2}^i)}{\Delta T^2} \quad (1) \\
V_t^k &= \frac{C_t^k - C_{t-1}^i}{\Delta T} + Acc_t^k \times \Delta T \quad (2) \\
\hat{C}_{t+1}^* &= C_t^k + V_t^k \times \Delta T + \frac{Acc_t^k \times (\Delta T)^2}{2} \quad (3)
\end{align*}
\]

Figure 1 depicts a typical prediction and score for a candidate. We define two “quality” scores for \(C_{t+1}^n\) in frame \(f_{t+1}\):

\[
S_1 = \log \left( \frac{d - mTh}{MTh - mTh} \right) \quad (4)
\]

\[
S_2 = \log \left( \frac{\theta}{180} \right) \quad (5)
\]

where \(d\) and \(\theta\) are the pixel distance and the angle between \(\hat{C}_{t+1}^*\) and \(C_{t+1}^n\), as shown in Fig. 1. Then the quality score for any particular path in the directed network is \(S_1 + S_2\) for all triplets inside the path. To avoid large negative scores for \(S_1\) and \(S_2\), we constrain their range to the interval \([S_m, 0]\) where \(S_m\) is predefined.

2.3. Token Transfer

To search for the optimal path, we propose a modified version of the Viterbi algorithm which we call Token Transfer. A Token holds two quantities, of which \(CNPath\) is the history of CNs that the Token has passed through, and \(Score\) is the accumulated score (i.e. the sum of \(S_1\) and \(S_2\) over all triplets in \(CNPath\)).

The longer a \(CNPath\) becomes, the more chance there is that it incorporates false ball candidates. We utilize a windowing technique to alleviate this problem. A Token Transfer process, shown in Algorithm 1, is executed afresh within each window (in our work, the window length and sliding step length are set to 21 and 5 empirically). Here, \(Toks_i\) represents a set of Tokens passed up to Candidate Node \(C_t^i\); \(tok_0\) is an initial Token whose \(Score=0\), \(CNPath=\phi\). In win-

Algorithm 1 Token Transfer

**Initialization:**
1. \(Toks_0 = \{tok_0\}\);

**Recursion:**
2. for all \(f_t\) within each window \(w_n\) do
3. if \(Toks_i == \phi\) then
4. \(Toks_i = \{tok_0\}\);
5. end if
6. \(Toks_{i+1} = \phi\);
7. for all \(tok \in Toks_i\) do
8. for all \(C_{t+1}^j \in C_{t+1}\) do
9. Calculate \(S_1\) and \(S_2\); \(S_1=0, S_2=0\) when \(t < 3\)
10. \(tok\rightarrow\text{Score} = tok\rightarrow\text{Score} + S_1 + S_2\);
11. \(tok\rightarrow\text{CNPath} = tok\rightarrow\text{CNPath} + C_{t+1}^j\);
12. Add \(tok\) into \(Toks_{i+1}\);
13. end for
14. end for
15. Pick the \(N\) top-ranked Tokens from \(Toks_{i+1}\) into \(Toks_{w_n}\);
16. end for

Now \(w_n\), we retain the \(N\) top-ranked Tokens for each frame (we used \(N = 2\)). The \(CNPath\) of these selected Tokens thus comprise a sub-trajectory set, \(Toks_{w_n}\), and are used as the input to the global layer.

3. GLOBAL LAYER: SUB-TRAJECTORIES SPLICE

At the global layer, our goal is to determine an optimal subset from the set of sub-trajectories \(Toks_{w_n}\), to form the final ball trajectory. To do this, we firstly construct a directed acyclic graph (DAG) based on these sub-trajectories, and then search for the optimal directed path through the DAG using Dynamic Programming.

3.1. Construction of Directed Acyclic Graph

We illustrate the kind of sub-trajectories we obtain in Fig. 2(a). Here, each column represents a window \((w_n)\) and contains a small number of sub-trajectories \((Tok_{w_n}^i)\). These sub-trajectories, also called Token Nodes (TNs), are then used to form a grid structure as shown in Fig. 2(b). The
DAG is constructed by linking TNs. Three different cases are possible, depending on the relationship between the two TNs (\(Tok^i_{w_n}\) and \(Tok^j_{w_{n+1}}\)) occurring in adjacent windows \(w_n\) and \(w_{n+1}\):

**Case 1. Temporal and spatial overlap:**
If the front portion of \(\text{CNPath}\) of \(Tok^i_{w_n}\) is the same as the rear portion of the \(\text{CNPath}\) of \(Tok^j_{w_{n+1}}\), the two TNs are connected with a directed arc from \(Tok^i_{w_n}\) to \(Tok^j_{w_{n+1}}\).

**Case 2. Temporal overlap only:**
There is no directed arc between \(Tok^i_{w_n}\) and \(Tok^j_{w_{n+1}}\).

**Case 3. No temporal overlap:**
For this case, we linearly extend \(Tok^i_{w_n}\) at its end point (in frame \(f_{t_1}\)) and linearly extend \(Tok^j_{w_{n+1}}\) at its start point (in frame \(f_{t_2}\)) along the direction of the velocity vectors of the two points. If a junction of the two extended lines exists and the distance between the two end points is less than \(MT\theta \times (t_2 - t_1)\), we connect the two TNs with a directed arc.

After connecting all tokens, we obtain a DAG, as shown in Fig. 2(c), in which the score of a TN in the \(i\)th window is defined as \(S(w_n, i)\). However, not every window contains true sub-trajectories: these windows are skipped. In Fig. 2(d), for each window, we append a Dummy Token Node (DTN) whose \(\text{CNPath} = \emptyset\) and \(\text{Score} = 0\). DTNs can connect to any TN and DTN, which we term Soft Linking.

![Diagram](image)

**Fig. 2.** Construction of DAG from (a) to (d) step by step.

### 3.2. Sub-Trajectories Splice

To form the optimal trajectory, we employ a Dynamic Programming approach to process the obtained DAG. The full algorithm is presented is presented in Algorithm 2. We define two recursive quantities \(F(w_n, i)\) and \(G(w_n)\). As illustrated in Fig. **?, \(F(w_n, i)\)**, computed only on TNs, indicates the minimum cost of the paths ending up at the \(i\)th TN in \(w_n\). \(G(w_n)\), computed only on DTNs, represents a set of costs of paths ending up at the DTN in \(w_n\). In addition, \(P(w_n, i)\) stores the backtracking information of the optimal path. Figure **?** gives a diagrammatic explanation of how to compute \(F(w_n, i)\) and \(G(w_n)\) recursively, and the algorithm is shown in Algorithm 2. In order to prevent a path from using only the dummy tokens when traversing the DAG, we set three conditions:

- **C1:** a path consisting only of dummy tokens is removed
- **C2:** a path containing three consecutive dummy tokens at the end is removed
- **C3:** the two top-ranking paths are retained in the recursion

We also assume that trajectories in which the direction changes very quickly are unlikely to be correct. Hence when searching through the DAG, we use a penalty factor \((pf)\) defined as:

\[
 pf = \frac{\text{Length of Partial Path}}{\text{Length of Partial Path} - 1}
\]

\(\text{Length of Partial Path}\) is found by examining the angle between each pair of consecutive path segments in each candidate path and noting the number of angles whose value is larger than a threshold (set to 70 degrees here). During searching for the optimal path, we weight \(F(w_n, i)\) by \(pf\)—hence the smaller the value of \(pf\) is, the more likely the path will be selected.

### 4. EVALUATION

#### 4.1. Experimental Setting

In our experiments, we used material from two tennis matches. One was a men’s singles match from the 2010 Australian Open and the other a men’s singles match from the 2011 US Open. We extracted 37206 and 24363 frames from the two match videos at a sampling rate of 25 frames per second. The ground truth was obtained by manually locating and storing the ball’s position in a frame. Frames in which the ball did not appear were labeled as inactive frames. After application of the candidate extraction schemes described in Section 2.1, the average number of detected ball candidates per frame was 4.5. The metric used for evaluating algorithms is the F1-score:

\[
F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

\[
\text{Precision} = \frac{n_{tp}}{n_{tp} + n_{fp}}
\]

\[
\text{Recall} = \frac{n_{tp}}{n_{tp} + n_{fn}}
\]

where:

- \(n_{tp}\) = # frames where a ball was detected and was present,
- \(n_{fp}\) = # frames where a ball was detected but wasn’t present,
- \(n_{tn}\) = # frames where a ball wasn’t detected and wasn’t present,
- \(n_{fn}\) = # frames where a ball wasn’t detected but was present.
Algorithm 2 Sub-Trajectories Splice

Initialization:
\[ G(w_1) = \phi, F(w_1, i) = S(w_1, i); \]

Recursion:
1. for all \( n = \{1, 2, ..., N\} \) do
2. \( \mathcal{G} \triangleq \{G(w_{n-1}) \mid \text{Condition1}\}; \)
3. \( F(w_n, i) = \min_j \{F(w_{n-1}, j) \cup \mathcal{G}\} + S(w_n, i) \times p_f; \)
4. \( G(w_n) = \{\text{Group} \mid \text{Condition2} \land \text{Condition3}\}; \)
5. \( P(w_n, i) = \arg \min_j \{F(w_n, i) \cup G(w_n)\}; \)
6. end for

Termination:
7. \( F^* = \min_i \{F(w_N, i) \cup G(w_N)\}; \)
8. \( P^* = \arg \min_i \{F(w_N, i) \cup G(w_N)\}; \)

In our experiments, the values of \( n_{tp}, n_{fp}, n_{tn}, \) and \( n_{fn} \) were obtained by comparing the distance between the actual ball position and the estimated ball position in each frame. We treat the hypothesised candidates as false candidates if this distance is larger than 10 pixels.

4.2. Evaluation

Figure 3 illustrates an example of ball tracking using our approach. Figure 3(a) plots the sub-trajectory set obtained in the local layer, while Fig. 3(b) shows the final estimated trajectory.

![Figure 3: A complete example of our approach.](image)

Table 1. The performance of three editions of method

<table>
<thead>
<tr>
<th>Data</th>
<th>Method</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>F1 (%)</th>
<th>TP Err. (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aust.</td>
<td>TL</td>
<td>72.01</td>
<td>72.50</td>
<td>72.26</td>
<td>1.94</td>
</tr>
<tr>
<td></td>
<td>TL+PF</td>
<td>73.13</td>
<td>73.11</td>
<td>73.12</td>
<td>1.94</td>
</tr>
<tr>
<td>U.S.</td>
<td>TL</td>
<td>81.47</td>
<td>67.27</td>
<td>73.69</td>
<td>1.97</td>
</tr>
<tr>
<td></td>
<td>TL+PF</td>
<td>82.31</td>
<td>67.52</td>
<td>74.19</td>
<td>1.79</td>
</tr>
</tbody>
</table>

false detections than the technique used in our previous work. 

5. CONCLUSION AND FURTHER WORK

In this paper, we have presented a two-layered data association method for tennis ball tracking in a complex scene. Experiments show that our approach is significantly more robust to false detections than the technique used in our previous work. In the future, we will extend the approach to integrate audio information. This will lay the foundations for making a high level analysis of game and making an analysis of player’s actions and tactics.

References


