A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match

F.Yan, W.Christmas and J.Kittler
Centre for Vision, Speech and Signal Processing
University of Surrey
Guildford, GU2 7XH, UK
{f.yan,w.christmas,j.kittler}@surrey.ac.uk

Abstract

Several tennis ball tracking algorithms have been reported in the literature. However, most of them use high quality video and multiple cameras, and the emphasis has been on coordinating the cameras, or visualising the tracking results. In this paper, we propose a tennis ball tracking algorithm for low quality off-air video recorded with a single camera. Multiple visual cues are exploited for tennis candidate detection. A particle filter with improved sampling efficiency is used to track the tennis candidates. Experimental results show that our algorithm is robust and has a tracking accuracy that is sufficiently high for automatic annotation of tennis matches.

1 Introduction

In automatic annotation of sports video, higher-level descriptions generally rely on low-level features. In the context of a tennis match, the evolution of the game is described by key events such as the tennis ball being hit, the tennis ball bouncing on the ground, etc. To detect these important events, the tracking of the tennis ball is required. However, tracking the small and fast moving tennis ball is a challenging task. Small objects usually have less features to detect, and are more vulnerable to distractions; the moving of the tennis ball is so fast that sometimes it is blurred into background; the tennis ball is also subject to occlusion, false detection, and sudden change of motion direction. This last point poses a major challenge for the tracking, especially when the ball is close to one of the players, where abrupt motion change can happen at the same time as occlusion and false detection.

Although various tracking algorithms have been developed in the past [1, 2, 4, 9], little has been seen in the literature on how robust tracking algorithms are used to track tennis ball. Most existing tennis systems use high quality video, which means simple algorithms such as blob tracker is sufficient [6, 7]. The emphasis has been on coordinating multiple cameras, or visualising the tracking results [5, 6, 7]. In this paper, we propose a robust tennis ball tracking algorithm for low quality off-air video recorded with a single camera. Multiple visual features are exploited for tennis candidate detection. A particle filter with improved sampling efficiency is used to track the tennis candidates. Experimental results

show that our algorithm is robust, and the tracking accuracy is high enough for automatic annotation.

This paper is organised as follows: Section 2 gives a brief introduction to our automatic tennis video annotation system. In Section 3, the methodology of the proposed tennis ball tracking algorithm is presented. Its performance is shown and discussed in Section 4. Conclusions are given in Section 5.

2 The Automatic Tennis Video Annotation System

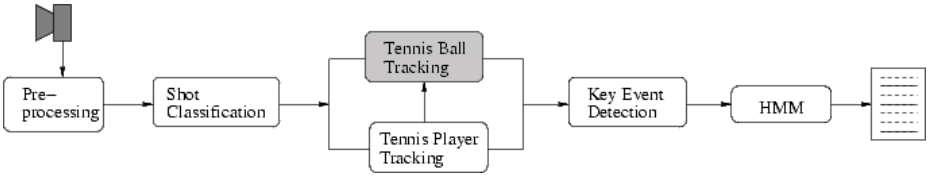


Figure 1: System diagram

Fig. 1 is the diagram of our automatic tennis annotation system. As a pre-processing stage, image frames are de-interlaced into fields, and the geometric distortion of camera lens is corrected. When the tennis ball is moving fast, it is alternately present and absent on successive frame lines, thus the need to operate on fields rather than frames. Shot boundaries are then detected and shots are classified into play shots and others (close-up, crowd etc.). For a play shot, the tennis players are tracked using standard background subtraction and a simple blob tracker. By examining the tennis ball trajectory provided by ball tracking module, and incorporating the player positions, key events (hit, bounce, etc.) are detected. Finally, a hidden Markov model is applied to the events to generate high level annotation, i.e., outcome of play, point, etc.

3 Methodology

In this section, we show how the difficulties in tennis ball tracking are tackled. Three major steps involved in the proposed tennis ball tracking algorithm are presented. First, foreground moving objects are extracted by differencing temporally neighbouring fields. Foreground blobs are then classified into tennis ball candidates and non-candidates using multiple visual cues. Finally, the candidates are tracked with a particle filter.

3.1 Motion Segmentation

Pixel-wise temporal differencing is adopted as means of motion detection. A pan-tilt-zoom camera is assumed. Homographies between fields are calculated by tracking corresponding corners, and then used to compensate the global motion between fields. Let f_k be the k^{th} field in a play shot, and f_l be a neighbouring field, which is already motion compensated with respect to f_k. For a pixel $p_k(m,n)$ in f_k with intensity $I_k(m,n)$ and indices m,n, a boolean variable is defined as

$$isFG_{k,l}(m,n) \triangleq [I_k(m,n) - I_l(m,n) > i_{th}]$$

where i_{th} is a threshold. Pixel $p_k(m,n)$ is classified as foreground pixel when

$$isFG_k(m,n) \triangleq \bigwedge_{\forall l \in L_k} \text{isFG}_{k,l}(m,n) = true$$

where Λ is the logical AND operation, and L_k is a collection of several neighbouring fields. In our implementation,

$$L_k = \{k-8, k-6, k-4, k+4, k+6, k+8\}$$

Note that f_{k-2} and f_{k+2} are not used, to prevent the ball being excluded when it is traveling at very low velocity.

The result of tennis player tracking is then incorporated. Pixels that belong to the players are excluded from foreground pixels.

3.2 Tennis Ball Candidate Detection

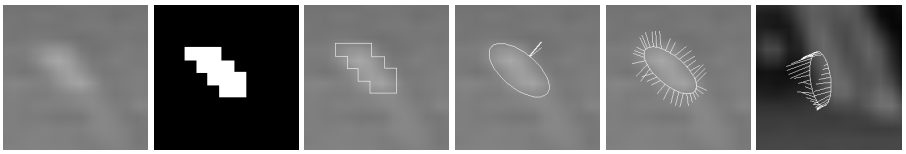
The foreground pixels obtained above are clustered into blobs according to their connectedness. A blob may be the true tennis ball, it may also be part of the player, the racket, or even part of the advertising boards, due to various inaccuracies. A method for foreground blob classification is required. In [6], the authors suggest that the tennis ball has a standardised yellow colour, which can be used as an important cue for the classification. However, in some off-air material, since the colour bandwidth is low, and the tennis ball has a very small size, its colour can be strongly affected by the colour of the background it is traveling on. Moreover, the video archives our system is targeted at are usually subject to artifacts introduced by analogue encoding, for instance, the PAL cross-colour effect, which appears as reddish noise colour "floating" in the image. This also suggests that colour information by itself is not sufficient for the classification. We suggest a multi-cue foreground blob classification method as follows.

For each foreground blob, a small surrounding area is included, and the enlarged blob is interpolated, using bi-linear interpolation. The corresponding binary image, in which a "true" stands for a foreground pixel, a "false" for a background pixel, is also interpolated. The edge pixels of the interpolated binary image are then extracted, and used as the edge of the blob in the interpolated intensity image. An ellipse is fitted to the edge pixels, according to a least square criterion. M points are then sampled along the ellipse. For each point, the normal direction is found, and the gradient is calculated using a 3×3 Sobel mask.

Fig. 2(a) - Fig. 2(d) shows the major steps involved in this process. The idea is to use the mean absolute angle difference of the normal direction and the gradient direction at all sample points for foreground blob classification. We define

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

where α_i is the absolute angle difference in radians at the i^{th} sample point. It takes a value ranging from 0 to π. Using the assumption that a target-originated blob is a local



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

Figure 2: An important feature for blob classification: absolute angle difference between gradient orientation and surface normal orientation. (a) interpolated intensity image, (b) interpolated binary image, (c) interpolated intensity image with blob edge, (d) fitted ellipse with surface normal (thick line) and Sobel gradient (thin line), (e) gradient vectors of an target-originated blob, $\alpha = 0.31$, (f) gradient vectors of a clutter-originated blob, $\alpha = 1.93$.

maximum in the intensity image and is approximately elliptical, a blob with smaller α is more likely to be target-originated.

An 8-dimensional feature vector is then constructed, with α as one dimension of it. Other dimensions include the coordinates of the blob center (row and column), the parameters of the fitted ellipse (major axis, minor axis), and the mean of the pixels inside the blob in HSV channels. An SVM is then trained to classify foreground blobs. Experimental results show that with this multi-cue blob classification method, a good accuracy can be achieved without imposing too much computational overhead.

3.3 Tennis Ball Tracking

Problem Modelling: Consider a linear time-invariant dynamic system defined by a system model and a observation model as follows:

$$\begin{aligned} \mathbf{x}_t &= A\mathbf{x}_{t-1} + \mathbf{v} \\ \mathbf{z}_t &= H\mathbf{x}_t + \mathbf{w} \end{aligned}$$

where $\mathbf{x}_t$ is the system state, $\mathbf{z}_t$ is the observation, and $\mathbf{v}$, $\mathbf{w}$ are process noise and observation noise, respectively. The objective is to estimate $\mathbf{x}_t$ recursively, using noise corrupted observation $\mathbf{z}_t$. In our tennis ball tracking problem, the position and velocity of the tennis ball in both vertical and horizontal directions are modelled by the state vector:

$$\mathbf{x}_t = (p_t^v,\ p_t^h,\ v_t^v,\ v_t^h)^T$$

Assuming the velocity of the tennis ball is constant in two successive fields, the state transition matrix A and observation matrix H are then determined as

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

The process noise $\mathbf{v}$ and observation noise $\mathbf{w}$ are further assumed to be zero-mean, white, and Gaussian random processes.

Tracking with a particle filter: Using the assumptions made above, we have:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \frac{1}{(2\pi)^2|Q|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{x}_t - A\mathbf{x}_{t-1})^T Q^{-1}(\mathbf{x}_t - A\mathbf{x}_{t-1})] \tag{1}$$

where Q is the covariance of the process noise.

Assuming the number of clutter-originated observations is Poisson distributed, the total observation density can be proved to be, up to proportionality,

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

where R is the covariance of the observation noise, $\mathbf{z}_t^j$ is the j^{th} tennis ball candidate, m_t is the number of candidates detected at time t, and β is a constant corresponding to the observation density caused by clutter [2].

According to Bayes' theorem, and using the result of Eq. (1) and Eq. (2), the posterior density $p(\mathbf{x}_t|Z_t)$ can be derived to be a Gaussian mixture. An optimal algorithm (in the Bayesian sense) for estimating this density has been proposed by Bar-Shalom and Forman in [1]. However, in their algorithm, the decomposition of densities with respect to all the observations is performed at each time step, leading to an exponentially increasing number of modes in the Gaussian mixture, and hence an exponentially increasing computational complexity. An elegant solution to this problem is provided by Monte Carlo simulation [8].

Assume at time $t-1$ the posterior density is represented by a set of particles with uniform weights, i.e.,

$$p(\mathbf{x}_{t-1}|Z_{t-1}) = \sum_{i=1}^{n} \frac{1}{n} \delta(\mathbf{x}_{t-1} - \mathbf{s}_{t-1}^i) \tag{3}$$

where n is the number of particles, $\delta(\cdot)$ is the Dirac delta function, and Z_{t-1} is the observation history up to time $t-1$. Using Eq. (3) and Eq. (1), the prior density at time t is obtained as:

$$p(\mathbf{x}_t|Z_{t-1}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(2\pi)^2|Q|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{x}_t - A\mathbf{s}_{t-1}^i)^T Q^{-1}(\mathbf{x}_t - A\mathbf{s}_{t-1}^i)] \tag{4}$$

According to Bayes' theorem, and using the result of Eq. (4) and Eq. (2), after some rearrangement, we get the posterior density at time t as:

$$p(\mathbf{x}_t|Z_t) \propto \sum_{i=1}^{n} \sum_{j=0}^{m_t} w_t^{i,j} \frac{1}{(2\pi)^2|C_t^{i,j}|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{x}_t - \mathbf{m}_t^{i,j})^T C_t^{i,j\,-1}(\mathbf{x}_t - \mathbf{m}_t^{i,j})] \tag{5}$$

where

$$w_t^{i,0} = \beta, \quad C_t^{i,0} = Q, \quad \mathbf{m}_t^{i,0} = A\mathbf{s}_{t-1}^i \tag{6}$$

for $j \neq 0$

$$w_t^{i,j} = \frac{1}{2\pi|R+HQH^T|^{\frac{1}{2}}} \exp[-\tfrac{1}{2}(\mathbf{z}_t^j - HA\mathbf{s}_{t-1}^i)^T (R+HQH^T)^{-1}(\mathbf{z}_t^j - HA\mathbf{s}_{t-1}^i)]$$

$$C_t^{i,j} = (I-KH)Q$$

$$\mathbf{m}_t^{i,j} = A\mathbf{s}_{t-1}^i + K(\mathbf{z}_t^j - HA\mathbf{s}_{t-1}^i) \tag{7}$$

and

$$K = QH^T(R + HQH^T)^{-1} \tag{8}$$

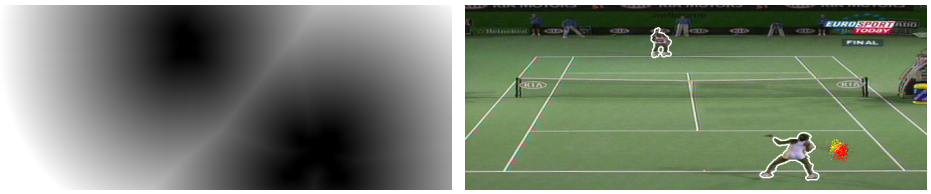
Eq. (5) to Eq. (8) show that the posterior density is a Gaussian mixture with $n \times (m_t + 1)$ components, and the weight, covariance, and mean of each component are analytically available. Moreover, it is straightforward to simulate a Gaussian mixture distribution. An update cycle of the particle filter is thus complete.

Two advantages are achieved with the proposed algorithm. Firstly, Monte Carlo simulation allows us, without losing Bayesian optimality, to limit the computational complexity by choosing the number of particles. Secondly, in the standard Sample-Importance-Resample particle filter [2], samples are drawn from prior probability distribution. When the likelihood is much more peaked than the prior, this leads to a very low sampling efficiency, since most particles will have negligible weights. Various techniques have been suggested to solve this problem, by "herding" the particles to the right part of the state space [8, 9]. In our case, however, those techniques are not needed. Since we can draw samples directly from the posterior density, our algorithm is optimal in terms of sampling efficiency [9].

Automatic switching between two dynamic models [3]: When the tennis ball is hit by a player, it changes its motion drastically. This may cause loss of tracking. To handle the abrupt motion change, when the tennis ball is close to a player, a second dynamic model with the following form is triggered with a certain probability:

$$A' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note that to use the second dynamic model, the only modification needed in the algorithm is to use A' instead of A in Eq. (6) and Eq. (7).



(a) Chamfer distance transform on the tennis player tracking result

(b) The second dynamic model is triggered when the ball is close to the players. Red points: particles propagated using the first dynamic model; yellow points: particles propagated using the second dynamic model

Figure 3: Automatic switching between two dynamic models

The probability that a particle is propagated with the second dynamic model is determined by the distance between the particle and the players, and also the "history" of the particle. The tennis players are tracked, and Chamfer distance transform is applied to the resulting binary image, as shown in Fig. 3(a). A threshold d_{th} is then set. A particle is said to be "close to a player" if $d_c < d_{th}$, where d_c is its corresponding Chamfer distance.

If a particle is not "close to a player", it is always propagated using the first model. If a particle $\mathbf{s}_t^\star$ is "close to a player", we search through its history backwards. Assume the first particle in its history that is *not* "close to a player" is encountered at time t_1, and the first particle that is propagated with the second model is encountered at t_2. If $t_2 < t_1$, $\mathbf{s}_t^\star$ is propagated using the second model with a probability of p', and using the first model with a probability of $1-p'$. If $t_2 > t_1$, the particle is always propagated using the first model.

Input: a set of particles at time $t-1$: $\{\mathbf{s}_{t-1}^i, \frac{1}{n}\}_{i=1}^n$
observations at time t: $\{\mathbf{z}_t^j\}_{j=1}^{m_t}$
Output: a new set of particles at time t: $\{\mathbf{s}_t^k, \frac{1}{n}\}_{k=1}^n$

- FOR (each particle $\mathbf{s}_{t-1}^i$ at time $t-1$)
  - find the corresponding Chamfer distance and the dynamic model to use
  - compute $w_t^{i,j}$, $C_t^{i,j}$, and $\mathbf{m}_t^{i,j}$ of each component in $p(\mathbf{x}_t|Z_t)$, using Eq. (6), Eq. (7) and appropriate dynamic model
  - normalise so that $\sum_{i=1}^n \sum_{j=0}^{m_t} w_t^{i,j} = 1$
  - store $w_t^{i,j}$, $C_t^{i,j}$, $\mathbf{m}_t^{i,j}$ and the index i
- END FOR
- FOR ($k = 1; k <= n; k++$)
  - select a pair (i,j) with probability $p(i,j) = w_t^{i,j}$. This can be done by cumulating the weight $w_t^{i,j}$ and using a random number normally distributed in $[0,1]$
  - sample from the $(i,j)^{th}$ component of the posterior density. i.e., the one with $C_t^{i,j}$ as covariance and $\mathbf{m}_t^{i,j}$ as mean.
  - store the sampled particle, along with the index of its parent particle, i
- END FOR

Table 1: Pseudo-code of one update cycle of the algorithm

Smoothing and observation origin identification: The accuracy of the tennis ball tracking result is critical for our application. It has great impact on the following event detection procedure, and will in turn affect the final annotation result. Unfortunately, estimates computed directly from $p(\mathbf{x}_t|Z_t)$ are usually not accurate enough. A smoothing process is required, which tries to estimate $p(\mathbf{x}_t|Z_T)$, i.e., the distribution of $\mathbf{x}_t$ in the light of all the observations in the sequence. A fast smoothing algorithm developed by Kitagawa [4] is adopted.

The smoothing procedure is straightforward. The set $\{S_t^i, \pi_t^i\}_{i=1}^n$ is stored at each time t instead of the set $\{\mathbf{s}_t^i, \pi_t^i\}_{i=1}^n$, where $S_t^i = (\mathbf{s}_1^{k_1^{i,t}}, \cdots, \mathbf{s}_t^{k_t^{i,t}})$ is the history of particle $\mathbf{s}_t^i$, $k_\tau^{i,t}$ is the index of the particle in S_t^i at time τ, and $k_t^{i,t} = i$. The smoothed density is then:

$$p(\mathbf{x}_t|Z_T) = \sum_{i=1}^n \pi_T^i \delta(\mathbf{x}_t - \mathbf{s}_t^{k_t^{i,T}}) \tag{9}$$

In our implementation, the weight π_t^i is always $\frac{1}{n}$. The only difference between the unsmoothed density $p(\mathbf{x}_t|Z_t)$ and the smoothed density $p(\mathbf{x}_t|Z_T)$ is that two different sets of

discrete support are used: if a particle, as a hypothesis, is reinforced "in the future", it will have more copies in the smoothed density.

Using Eq. (9) and the observation model, we get:

$$p(\mathbf{z}_t|Z_T) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{(2\pi)|R|^{\frac{1}{2}}} \exp[-\frac{1}{2}(\mathbf{z}_t - H\mathbf{s}_t^{k_t^{i,T}})^T R^{-1}(\mathbf{z}_t - H\mathbf{s}_t^{k_t^{i,T}})] \tag{10}$$

Substituting $\mathbf{z}_t^j, j = 1, \cdots, m_t$ into Eq. (10) we get the likelihood of each observation. If the likelihood of at least one observation is greater than a threshold l_{th}, the observation with highest likelihood is used as the final tennis ball position. If none of the likelihoods is greater than l_{th}, we assume the tennis ball is not detected, and its position is interpolated. This extra process identifies the origin of each observation, so the effect of outliers is completely eliminated.

4 Experiments

To evaluate the performances of the proposed tennis ball tracking algorithms, experiments were carried out on video sequences from the 2003 Australian Open tennis tournament. We ran our tracking algorithm on 70 shots that were classified as "play" shots by our system (Fig. 1). In total the 70 shots are approximately 8 minutes long, and contain 23096 fields. In the experiments, a particle number of $n = 1000$ is used, and the standard deviations of observation noise and process noise are 0.5 pixel and 2 pixels, respectively.

For this quantity of data, it is not feasible to provide ground truth. However, good results were observed in visual inspection. On all 70 shots, tracking is successfully initialised using a simple method, which detects smooth motions. The tracking result is excellent on 69 shots: when the true tennis ball is detected as one of the candidates, it is always correctly identified as the target-originated observation; in the rare cases where, due to occlusion or errors of blob classification, the tennis ball is not detected as one of the candidates, its position is accurately interpolated. Examples of the tracking result are shown in Fig. 6.

The only tracking failure happens in a shot containing 466 fields, where the tennis ball is either occluded by the "near player" or confused with the sideline in 34 successive fields, before it reappears at the far side of the court. Tracking is lost during this period. A tracking re-initialization mechanism is still to be developed.

5 Conclusions

In this paper, we present a tennis ball tracking algorithm for low quality off-air video. Multiple visual features are exploited for foreground blob classification, among which a novel feature for detecting elliptical objects is proposed. A particle filter is used to track the tennis candidates. Samples are drawn directly from the posterior density. As a result, an improved sampling efficiency is achieved. Smoothing and observation origin identification are then used to refine the trajectory, to give higher tracking accuracy. Experimental results show that our algorithm is robust and has an accuracy that is sufficiently high for tennis annotation.

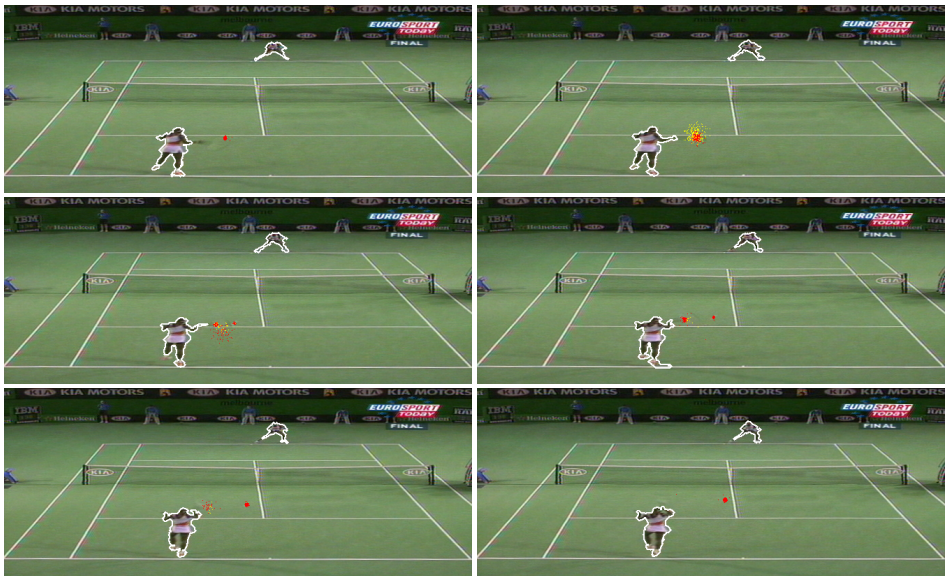


Figure 4: The particle filter in operation. 6 successive fields are shown, the field numbers are (from left to right, top to bottom) 140-145. 140: particles are closely clustered around the observation, this is the result of sampling directly from posterior density, and indicates a high sampling efficiency. 141: when no tennis ball candidate is detected, the particles are governed only by the prediction model. 142-145: when multiple observations are made, ambiguity arises. The particle filter tracks each of the observations till the ambiguity is resolved. Particle colours are as in Fig. 3(b)

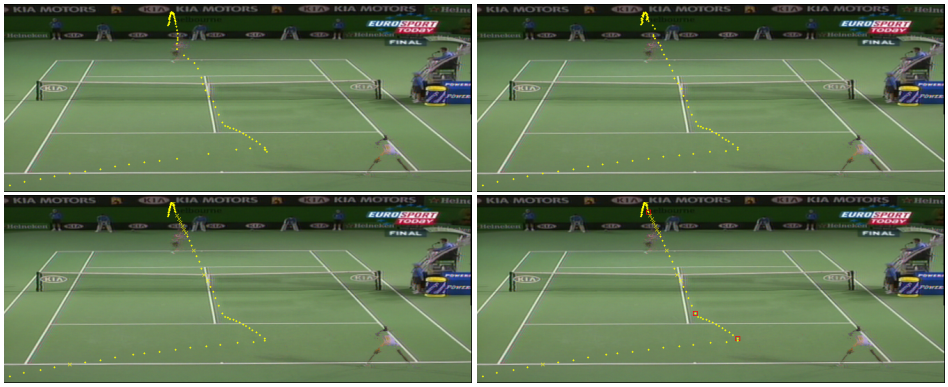


Figure 5: Smoothing and observation origin identification. Trajectories are shown on a background image constructed with image mosaicing technique. Top left: Mean of $p(\mathbf{x}_t|Z_t)$ (unsmoothed density). Top right: Mean of $p(\mathbf{x}_t|Z_T)$ (smoothed density). Bottom left: Tennis ball trajectory after observation origin identification. Points: observations; crosses: interpolated positions. Bottom right: The trajectory is now accurate enough for key events (hit, bounce, etc.) detection. Red square: key events detected using motion discontinuity in the trajectory.

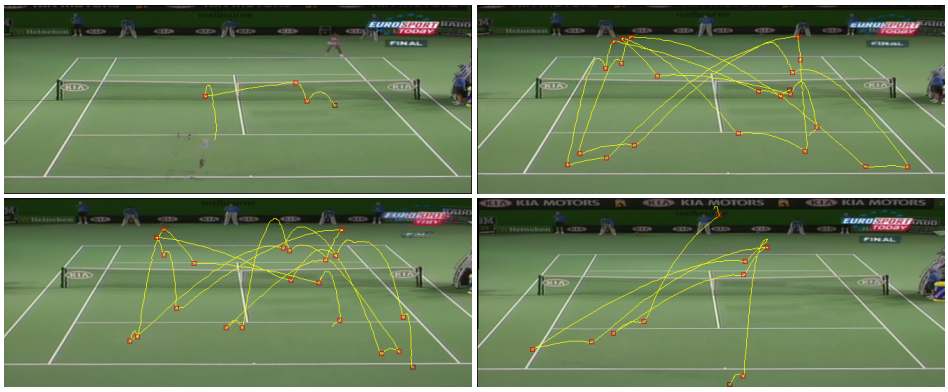


Figure 6: Some examples of the tracking result. Yellow lines: reconstructed ball trajectories. Red squares: detected key events.

Acknowledgements

This work has been supported by the EU IST-2001-34401 Project VAMPIRE.

References

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

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

[3] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *IEEE International Conference on Computer Vision*, pages 107–112, 1998.

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

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

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

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

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

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