Automatic Annotation of Tennis Game: An Integration of Audio, Vision, and Learning

Fei Yan[a,1], Josef Kittler[a], David Windridge[a], William Christmas[a], Krystian Mikolajczyk[a], Stephen Cox[b], Qiang Huang[b]

[a]*Centre for Vision, Speech, and Signal Processing, University of Surrey, Guildford, United Kingdom, GU2 7XH.*
[b]*School of Computing Sciences, University of East Anglia, Norwich, United Kingdom, NR4 7TJ.*

Abstract

Fully automatic annotation of tennis game using broadcast video is a task with a great potential but enormous challenges. In this paper we describe our approach to this task, which integrates computer vision, machine listening, and machine learning. At the low level processing, we improve upon our previously proposed state-of-the-art tennis ball tracking algorithm and employ audio signal processing techniques to detect key events and construct features for classifying the events. At the high level analysis, we model event classification as a sequence labelling problem, and investigate four machine learning techniques using simulated event sequences. Finally, we evaluate our proposed approach on three real world tennis games, and discuss the interplay between audio, vision and learning. To the best of our knowledge, our system is the only one that can annotate tennis game at such a detailed level.

Keywords: Tennis annotation, Object tracking, Audio event classification,

[1]Email of the corresponding author: *f.yan@surrey.ac.uk* (Fei Yan).

Sequence labelling, Structured output learning, Hidden Markov model

1. Introduction

The rapid growth of sports video databases demands effective and efficient tools for automatic annotation. Owing to advances in computer vision, signal processing, and machine learning, building such tools has become possible [1, 2, 3]. Such annotation systems have many potential applications, e.g., content-based video retrieval, enhanced broadcast, summarisation, object-based video encoding, automatic analysis of player tactics, to name a few.

Much of the effort in sports video annotation has been devoted to court games such as tennis and badminton, not only due to their popularity, but also to the fact that court games have well structured rules. A court game usually involves two (or two groups of) players hitting a ball alternately. A point is awarded when the ball fails to travel over a net or lands outside a court area. The task of court game annotation then consists in following the evolution of a game in terms of a sequence of key events, such as serve, ball bouncing on the ground, player hitting the ball, ball hitting the net, etc.

On the other hand, building a fully automatic annotation system for broadcast tennis video is an extremely challenging task. Unlike existing commercial systems such as the Hawk-Eye [4], which use multiple calibrated high-speed cameras, broadcast video archives recorded with a monocular camera pose great difficulties to the annotation. These difficulties include: video encoding artifacts, frame-dropping due to transmission problems, illumination changes in outdoor games, acoustic mismatch between tournaments, frequent switching between different types of shots, and special effects and banners/logos inserted by the broadcaster, to name a few. As a result of the challenges, most existing tennis applications focus only on a specific aspect of the annotation problem, e.g., ball tracking [5, 6], action recognition [7]; or only annotate at a crude level, e.g., highlight detection [3], shot type classification [1]. Moreover, they are typically evaluated on small datasets with a few thousands of frames [5, 6, 7].

In this paper, we propose a comprehensive approach to automatic annotation of tennis games, by integrating computer vision, audio signal processing, and machine learning. We define the problem our system tackles as follows:

- Input: a broadcast tennis video without any manual pre-processing and pre-filtering, that is, the video typically contains various types of shots, e.g. play, close-up, crowd, and commercial;

- Output: ball event detection: 3D (row and column of frame + frame number) coordinates of where the ball changes its motion; and ball event classification: the nature of detected ball events in terms of five distinct event labels: serve, hit, bounce, net, and null, which corresponds to erroneous event detection.

To the best of our knowledge, our system is the only one that can annotate at such a detailed level.

To achieve the goal defined above, at the feature level, we improve upon our previous work and propose a ball tracking algorithm that works in the more cluttered and therefore more challenging tennis doubles games. The identified ball trajectories are used for event detection and as one feature for event classification. A second feature for classification is extracted by audio signal processing. At the learning level, we model event classification as a sequence labelling problem. We investigate four representative learning techniques and identify their advantages on simulated event sequences. Finally, our approach is evaluated on three real world broadcast tennis videos containing hundreds of thousands of frames. Discussions on the interplay between audio, vision, and learning are also provided. Note that this paper extends our preliminary work [8] by including the construction of visual and audio features, the integration of visual and audio modalities at the learning level, and a more comprehensive investigation of learning techniques.

The rest of this paper is organised as follows. Section 2 gives an overview of the proposed approach. The construction of features, including visual and audio features, is described in Section 3. Four learning techniques are then reviewed and compared on simulated event sequences in Section 4. Results on real world tennis games and discussions on the results are provided in Section 5. Finally Section 6 concludes the paper.

2. Overview of Our Approach

A diagram of our proposed system is illustrated in Fig. 1. We assume a tennis video recorded with a monocular and static camera, e.g., a broadcast tennis video. If the video is interlaced, its frames are first de-interlaced into fields, in order to alleviate the effects of temporal aliasing. For the sake of simplicity, in the remainder of this paper, we will use “frames” to refer to both frames of progressive videos and fields of interlaced videos. After de-interlacing, the geometric distortion of camera lens is corrected. De-interlacing and geometric correction are considered “pre-processing” and

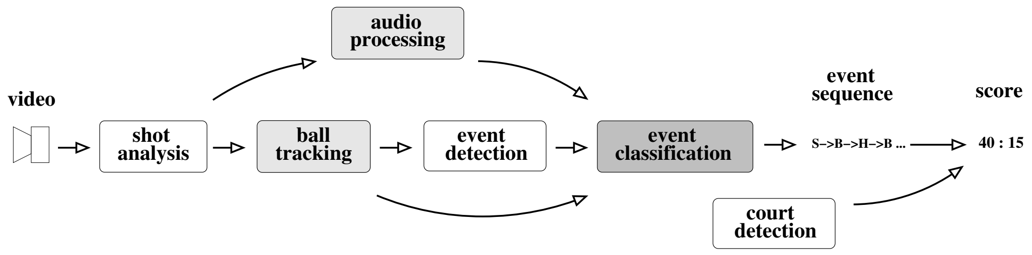


Figure 1: System diagram of our proposed tennis video annotation approach. The light-shaded blocks are covers in Sections 3.1 and 3.2 respectively; the dark-shaded block is covered in Section 4.

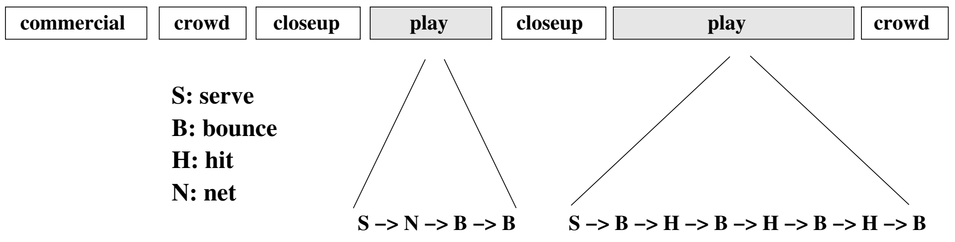


Figure 2: Typical composition of broadcast tennis videos and two examples of typical sequences of events in play shots.

omitted from Fig. 1.

A broadcast tennis video is typically composed of different types of shots, such as play, close-up, crowd, and commercial. In the “shot analysis” block of Fig. 1, shot boundaries are detected using colour histogram intersection between adjacent frames. Shots are then classified into appropriate types using a combination of colour histogram mode and corner point continuity [9]. An example of the composition of a broadcast tennis video is shown in Fig. 2, where two examples of typical sequences of events in tennis are also given. The first example corresponds to a failed serve: the serve is followed by a net, then by two bounces under the net. The second example contains a short rally, producing a sequence of alternate bounces and hits.

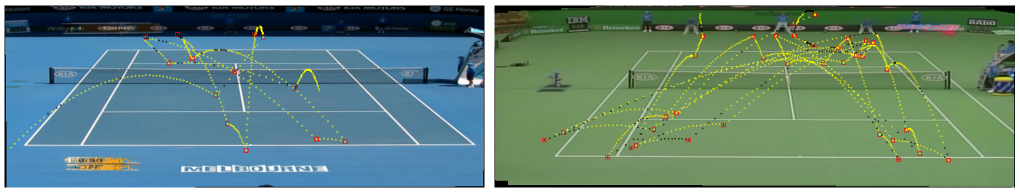


Figure 3: Two examples of ball tracking results with ball event detection. Yellow dots: detected ball positions. Black dots: interpolated ball positions. Red squares: detected ball events. Note that there are erroneous event detections in both examples. Note also that in the plots the ball trajectories are superimposed on the "mosaic" image, where moving objects such as players have been removed.

For a play shot, the ball is tracked using a combination of computer vision and data association techniques, which we will describe in more detail in Section 3.1. By examining the tracked ball trajectories, motion discontinuity points are *detected* as "key events". Two examples of ball tracking and event detection results are shown in Fig. 3, where each key event is denoted by a red square. The detected events are then *classified* into five types: serve, bounce, hit, net, and "null", which is caused by erroneous event detection. Two features are exploited for this classification task: information extracted from ball trajectories, i.e. location, velocity and acceleration around the events (Section 3.1); and audio event likelihoods from audio processing (Section 3.2).

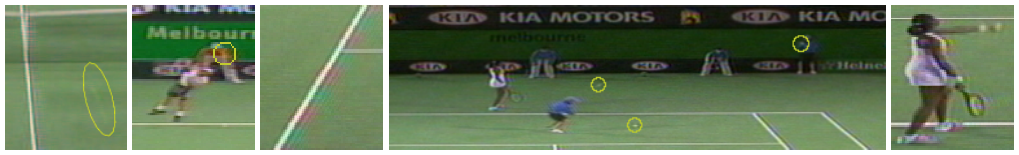
In addition to the features, the temporal correlations induced by tennis rules should also be exploited for classifying the events. For instance, a serve is likely to be followed by a bounce or a net, while a net almost certainly by a bounce. The focus of the "event classification" block of Fig. 1 is combining observations (features) and temporal correlations to achieve optimal classification accuracy. We model event classification as a sequence labelling problem, and provide an evaluation of several learning techniques on simulated event sequences in Section 4.

3. Extraction of Audio and Visual Features

In this section, we first introduce a ball tracking algorithm which improves upon our previous work. We sacrifice completeness for conciseness, and give an outline of the complete algorithm and discuss in detail only the modifications. Interested readers are referred to [10] for details of the complete algorithm. The tracked ball trajectories are used for event detection and also as a feature for event classification. In the second half of this section, we describe the second feature for event classification that is based on audio processing.

3.1. Ball tracking

Ball trajectories carry rich semantic information and play a central role in court game understanding. However, tracking a ball in broadcast video is an extremely challenging task. In fact, most of the existing court game annotation systems avoid ball tracking and rely only on audio and player information [11, 12, 13, 3, 14]. In broadcast videos the ball can occupy as few as only 5 pixels; it can travel at very high speed and blur into the background; the ball is also subject to occlusion and sudden change of motion direction. Furthermore, motion blur, occlusion, and abrupt motion change tend to occur together when the ball is close to one of the players. Example images demonstrating the challenges in tennis ball tracking are shown in Fig. 4.



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

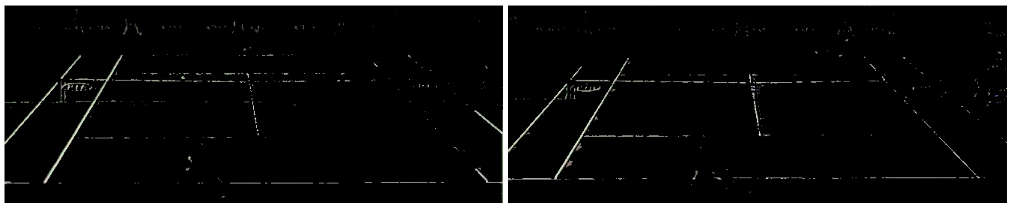
Figure 4: Image patches cropped from frames demonstrating the challenges of tennis ball tracking. (a) Ball travels at high speed and blurs into background, making it very hard to detect. (b) Far player serves. The ball region contains only a few pixels, and due to low bandwidth of colour channel its colour is strongly affected by the background colour. (c) Chroma noise caused by PAL cross-colour effect. This may introduce false ball candidates along the court lines. (d) Multiple balls in one frame. A new ball (left) is thrown in by a ball boy while the ball used for play (middle) is still in the scene. There is another ball (right) in the ball boy's hand. (e) A wristband can look very similar to the ball, and can form a smooth trajectory as the player strikes the ball.

To tackle these difficulties, we improve upon a ball tracking algorithm we proposed previously [10]. The operations of the algorithm in [10] can be summarised as follows[2]:

1. The camera position is assumed fixed, and the global transformation between frames is assumed to be a homography [15]. The homography is found by: tracking corners through the sequence; applying RANSAC to the corners to find a robust estimate of the homography; and finally, applying a Levenberg-Marquardt optimiser [9, 16].

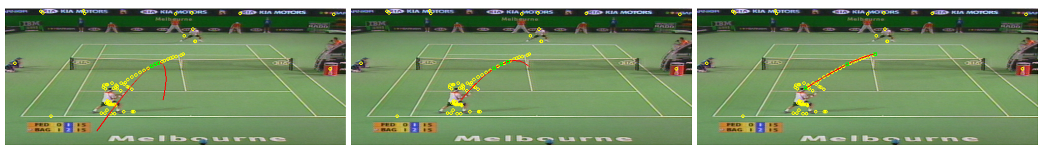
2. The global motion between frames is compensated for using the estimated homography. Foreground moving blobs are found by temporal

[2]A video file "ball-tracking.avi" is submitted with this manuscript to demonstrate this algorithm.



(a) (b)

Figure 5: Two example images of the output of temporal differencing and morphological opening. Due to noise in original frames, motion of players, and inaccuracy in homography computation, these residual maps are also noisy. The blobs are to be classified into ball candidates and not-ball candidates.



(a) (b) (c)

Figure 6: Model fitting and model optimisation. Yellow circles: ball candidates inside an interval of 31 frames ($V = 15$). Green squares: candidate triplet used for model fitting. Red curve: fitted dynamic model. (a) Fitting a dynamic model to the seed triplet. (b) and (c): the first and third iterations of model optimisation. Convergence is achieved after the third iteration.

differencing of successive frames, followed by a morphological opening operator which helps remove noise. Two example images of the output of this step are shown in Fig. 5.

3. Moving blobs are then classified as ball candidates and not-ball candidates using their size, shape and gradient direction at blob boundary.

4. A temporal sliding window is considered which centres at frame i and spans frame $i-V$ to frame $i+V$. For each candidate in frame i, we search in a small ellipsoid around it in the column-row-time space for one candidate from frame $i-1$ and one candidate from frame $i+1$. If such candidates exist, we call the three candidates inside the ellipsoid a "seed triplet", and fit a constant acceleration dynamic model.

5. Candidates within the sliding window that are consistent with the fitted model are identified as "supports" of the model, and the model is refined recursively using new triplets selected from the supports. This is done in a greedy fashion until convergence, i.e., when the "cost" λ of the model does not decrease any more. λ is defined as:

   $$\lambda = \sum_{j=i-V}^{i+V} \sum_{k} \rho(\mathbf{p}_j^k) \tag{1}$$

   with the per-candidate cost:

   $$\rho(\mathbf{p}_j^k) = \begin{cases} d^2(\hat{\mathbf{p}}_j, \mathbf{p}_j^k) & \text{if } d(\hat{\mathbf{p}}_j, \mathbf{p}_j^k) < d_{th} \\ d_{th}^2 & \text{if } d(\hat{\mathbf{p}}_j, \mathbf{p}_j^k) \geq d_{th} \end{cases} \tag{2}$$

   where $\mathbf{p}_j^k$ is the observed position of the k^{th} ball candidate in frame j, $\hat{\mathbf{p}}_j$ is the estimated ball position in frame j as given by the current model, $d(\cdot, \cdot)$ is the Euclidean distance, and d_{th} is a predefined threshold. A converged dynamic model together with its supports is called a "tracklet" and corresponds to an interval when the ball is in free flight. An example of the model fitting/optimisation process is given in Fig. 6.

6. As the sliding window moves, a sequence of tracklets is generated. These tracklets may have originated from the ball or from clutter. A weighted and directed 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 ball trajectories are obtained by computing the shortest paths between all pairs of nodes (tracklets) and analysing the paths.

The tracking algorithm summarised above works well in real world broadcast tennis videos of singles games. On the other hand, doubles games are considerably more challenging: the doubled number of players means more clutter, more occlusion, and more abrupt motion change. In order to cope with the increased challenges, we modified steps 4 and 5 of the previous algorithm to get more accurate tracklets.

First, we modify the way a dynamic model is computed. In [10] a constant acceleration model is solved exactly for three candidates in three frames: in the first iteration for the seed triplet, and in subsequent iterations for the three supports that are temporally maximally apart from each other. This scheme has two disadvantages: 1) It assumes the three observed ball candidate positions are noise-free, which is never the case in reality. 2) It uses only three candidates for computing the model, ignoring a potentially large number of supports. To remedy these problems, we assume the observed candidate positions are corrupted by a zero mean Gaussian noise, and jointly estimate the true positions and solve the dynamic model with a least squares (LS) estimator.

More specifically, let $\mathcal{J}$ be a set of frame numbers such that for each $j \in \mathcal{J}$ a candidate in frame j is used for computing the dynamic model. For example, in the first iteration, $\mathcal{J}$ contains the numbers of frames from which the seed triplet is drawn. Let $\{\mathbf{p}_j = (r_j, c_j)^T\}_{j \in \mathcal{J}}$ be the set of *observed* positions of candidates in terms of row and column, $\{\hat{\mathbf{p}}_j = (\hat{r}_j, \hat{c}_j)^T\}_{j \in \mathcal{J}}$ be the set of corresponding *estimated* true positions. Also let $\hat{\mathbf{p}}_i$ be the estimated position in frame i (middle frame of the sliding window), $\hat{\mathbf{v}}_i$ be the estimated velocity of the ball in frame i, and $\hat{\mathbf{a}}$ be the estimated acceleration, which is assumed constant for the dynamic model. Then we have:

$$\hat{\mathbf{p}}_j = \hat{\mathbf{p}}_i + (j - i)\hat{\mathbf{v}}_i + \frac{1}{2}(j - i)^2 \hat{\mathbf{a}} \odot \hat{\mathbf{a}} \quad \forall j \in \mathcal{J} \tag{3}$$

where $\odot$ denotes the element-wise multiplication. $\hat{\mathbf{p}}_i$, $\hat{\mathbf{v}}_i$, and $\hat{\mathbf{a}}$ can then be estimated by solving the LS problem:

$$\min_{\hat{\mathbf{p}}_i, \hat{\mathbf{v}}_i, \hat{\mathbf{a}}} \sum_{j \in \mathcal{J}} ||\mathbf{p}_j - \hat{\mathbf{p}}_j||^2 \tag{4}$$

Compared to the model fitting scheme in [10], the LS estimator in Eq. (4) does not assume noise-free observations, and does not impose the constraint that $|\mathcal{J}| = 3$. As a result, it leads to a more stable estimation of dynamic models.

We also modify the way candidates for model computation are selected in each iteration. In [10], $|\mathcal{J}|$ is set to 3, and the three candidates that are in the support set of the current model and are temporally maximally apart

Algorithm 1 Recursively refine the dynamic model for a seed triplet.

Input: All candidates in frames $i - V$ to $i + V$; a seed triplet in frames $i, i-1, i+1$; the number of random samples in each iteration R; the size of each random sample $|\mathcal{J}|$.

Output: A converged dynamic model $\mathcal{M}$ and associated set of supports $\mathcal{S}$, i.e., a tracklet.

```
 1: Fit a dynamic model $\mathcal{M}_0$ to the seed triplet using Eq. (4), identify the set of supports
    $\mathcal{S}_0$, compute the cost $\lambda_0$ of $\mathcal{M}_0$, and set $t = 1$;
 2: repeat
 3:     Set $\lambda_t = \infty$;
 4:     for $r = 1, \cdots, R$ do
 5:         Randomly sample $|\mathcal{J}| \geq 3$ supports from $\mathcal{M}_{t-1}$;
 6:         Fit a dynamic model $\mathcal{M}_{t,r}$ to the random sample using Eq. (4), identify the set
            of supports $\mathcal{S}_{t,r}$, compute the cost $\lambda_{t,r}$ of $\mathcal{M}_{t,r}$;
 7:         if $\lambda_{t,r} < \lambda_t$ then
 8:             Set $\lambda_t = \lambda_{t,r}$, $\mathcal{M} = \mathcal{M}_{t,r}$ and $\mathcal{S} = \mathcal{S}_{t,r}$;
 9:         end if
10:     end for
11:     Set $t = t + 1$;
12: until $\lambda_{t-1} \geq \lambda_{t-2}$
```

from each other are selected for refining the dynamic model. The idea is that interpolation in general is more reliable than extrapolation, and that this greedy strategy leads to quick convergence. However, in practice, greedy search can get stuck in local minima, degrading significantly the quality of the estimated dynamic model.

To remedy this, we introduce a randomised process inside each iteration. This randomisation greatly increases the chance of escaping from local minima, and as a result leads to more accurate model computation. Details of the improved algorithm for refining a dynamic model are provided in Algorithm 1. Note that the size of each random sample in Algorithm 1, $|\mathcal{J}|$, is not limited to 3 any more, thanks to the new LS formulation in Eq. (4).

In our experiments, the number of inner random samples is set to $R = 100$, which is larger than the number of iterations taken for the greedy search to converge (typically less than 10). The LS problem in Eq. (4) is also more expensive than the exact solution in [10]. However, in practice we observe that the two modifications do not significantly slow down the model fitting/refining process. With a frame rate at 25 frames per second, model fitting/refining runs comfortably in real time.

After ball tracking, key events are identified by detecting motion discontinuity in the ball trajectory [10]. Recall that each red square in Fig. 3 corresponds to one detected key event. The task of tennis annotation then consists in classifying the key events into five types: serve, bounce, hit, net, and "null" which corresponds to false positives in event detection. The ball dynamics around a detected event are used as one feature for event classification. More specifically, for a detected event in frame i, let $\hat{\mathbf{p}}_{i-1}$, $\hat{\mathbf{p}}_i$, $\hat{\mathbf{p}}_{i+1}$ be the estimated ball positions in frames $i-1, i, i+1$, $\hat{\mathbf{v}}_{i-1}$, $\hat{\mathbf{v}}_i$ be the estimated velocities, and $\hat{\mathbf{a}}_i$ be the estimated acceleration. We also compute the magnitude and orientation $\hat{\mathbf{a}}_i^{\dagger}$ of $\hat{\mathbf{a}}_i$. The 14 dimensional feature that is based on ball trajectory is finally defined as:

$$\mathbf{x}^{traj} = (\hat{\mathbf{p}}_{i-1}^T, \hat{\mathbf{p}}_i^T, \hat{\mathbf{p}}_{i+1}^T, \hat{\mathbf{v}}_{i-1}^T, \hat{\mathbf{v}}_i^T, \hat{\mathbf{a}}_i^T, \hat{\mathbf{a}}_i^{\dagger T})^T \tag{5}$$

3.2. Audio processing

In order to extract audio features for event classification, seven types of audio events are defined [17], as summarised in Table 1. Note that the set of audio events do not completely overlap with the four events for annotation (serve, bounce, hit, net), as some of the events for annotation e.g. bounce do not produce a characteristic and clearly audible sound, especially in the presence of crowd noise.

Table 1: Definition of seven audio events

index j	audio event	description
1	umpire	chair umpire's speech, e.g. reporting score
2	line judge	line judge's shout, e.g., reporting serve out, fault
3	hit	this corresponds to the "serve" and "hit" events for annotation
4	crowd	crowd noise, e.g. applause
5	beep	beep sound signalling e.g. let
6	commentator	commentators' speech
7	silence	silence

A men's singles game from Wimbledon Open 2008 is used for building models of the audio events. We first segment the sound track into audio frames of 30ms in length with 20ms overlap between consecutive frames. As a result, the audio frame rate is at 100 frames per second, which is higher than the video frame rate (50 for interlaced videos and 25 for progressive ones). Extra care is taken to ensure synchronisation between audio and video frames. The audio frames are labelled in terms of the seven audio events. We compute 39 dimensional Mel-frequency cepstral coefficients (MFCCs) for each frame and build a Gaussian mixture model for each audio event type.

Once the generative models are built, a test audio frame could be classified using straightforward maximum likelihood (ML). However, characteristics of audio events may vary across games and tournaments, significantly degrading the performance of the ML estimator. In order to reduce the impact of acoustic mismatches between training and test data, we employ a confidence measure. Let L_i^j be the log likelihood of audio frame i being the audio event j, where $j = 1, \cdots, 7$ correspond to the seven audio events, and L_i^j is given by the j^{th} Gaussian mixture model. The confidence measure of audio frame i being audio event j is then defined as:

$$D_i^j = L_i^j - \max_{k \neq j} L_i^k \tag{6}$$

Using the difference between log likelihoods provides some immunity to mismatch between training and test distributions: the mismatch will, to a certain extent, be cancelled by the differencing operation.

For a detected event in audio frame i, we wish to compute some statistics of the confidence measures in a neighbourhood of i and use it as a feature for event classification. In practice, we find that the *max* operator performs the best. Because our interest is in the "hit" audio event, we use only the confidence measure associated with this event. As a result, the audio-based feature for event classification is one dimensional:

$$\mathbf{x}^{audio} = \max_{i-W \le l \le i+W} D_l^3 \tag{7}$$

where the superscript $j = 3$ corresponds to audio event "hit", and W is set to 10 in our experiments. Note that event detection and the computation of the trajectory based features Eq. (5) are both done in terms of video frames. To ensure synchronisation, one could "map" audio frames to video frames by dividing i, l, W in Eq. (7) by 2 (for interlaced video) or 4 (for progressive video).

4. Learning Techniques for Event Classification

In the previous section, we have detected key events, and extracted features for each detected event. The two types of features extracted can be combined e.g. using vector concatenation $\mathbf{x} = (\mathbf{x}^{traj\,T}, \mathbf{x}^{audio\,T})^T$. For a given play shot, suppose a sequence of o events are detected, i.e., there are o "tokens" in the sequence. Let $y^s, s = 1, \cdots, o$ be the (micro-)label of the s^{th} event. y^s can take any value in the set of $\{serve, bounce, hit, net, null\}$, where $null$ is the label for false positives in event detection. The overall (structured) label of the play shot is then composed of a sequence of micro-labels:

$$y = y^1 \to y^2 \to \cdots \to y^o \tag{8}$$

Similarly let $\mathbf{x}^s, s = 1, \cdots, o$ be the concatenated feature for the s^{th} event. The overall (structured) feature of the play shot is then:

$$x = \mathbf{x}^1 \to \mathbf{x}^2 \to \cdots \to \mathbf{x}^o \tag{9}$$

In Eq. (8) and Eq. (9) we have used "$\rightarrow$" to indicate the sequential nature of the label y and the feature x. It is now clear that we have a structured learning problem: given a training set of feature/label pairs $\{x_i, y_i\}_{i=1}^m$, where $x_i = \mathbf{x}_i^1 \rightarrow \mathbf{x}_i^2 \rightarrow \cdots \rightarrow \mathbf{x}_i^{o_i}$, $y_i = y_i^1 \rightarrow y_i^2 \rightarrow \cdots \rightarrow y_i^{o_i}$, and o_i is the number of events (or tokens) of play shot i in the training set, we want to learn a structured classifier that can assign a label y to a test pattern x.

For such a sequence labelling task, we could ignore the internal structure of the labels and learn non-structured classifiers such as naive Bayes or SVM to assign micro-labels to individual events. On the other hand, as structured methods hidden Markov model (HMM) and structured SVM (S-SVM) can exploit the dependency of micro-labels. Among the four learning methods mentioned, naive Bayes, SVM and HMM are very well known. In the following we briefly review structured SVM. We will then evaluate the four methods on simulated event sequences. Through the simulation, the relative advantages of the different learning techniques are identified. A taxonomy of four learning techniques is given in Table 2.

Table 2: Taxonomy of learning techniques applicable to sequence labelling.

	Non-Structured	Structured
Generative	naive Bayes	Hidden Markov Model (HMM)
Discriminative	Support Vector Machine (SVM)	Structured SVM (S-SVM)

Structured SVM. In a nutshell, structured output learning (SOL) jointly embeds input-output pairs (x_i, y_i) into a feature space, and applies linear classifiers in the feature space. In the case of hinge loss, a max-margin hyperplane is sought, and the resulting learning machine can be thought of as structured SVM (S-SVM) [18]. More specifically, we assume for any training example (x_i, y_i),

$$\mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq \Delta(y_i, y),\ \forall y \in \mathcal{Y}\backslash y_i$$

where $\phi(\cdot,\cdot)$ is the joint embedding, $\Delta(\cdot,\cdot)$ is a label loss function measuring the distance between two labels, and $\mathcal{Y}$ is the set of all possible structured labels. Introducing regularisation and slack variables ξ_i, the max-margin hyperplane $\mathbf{w}^*$ is found by solving:

$$\begin{gathered}\min_{\mathbf{w}} \tfrac{1}{2}||\mathbf{w}||^2 + C\textstyle\sum_{i=1}^m \xi_i \\ \text{s.t.}\mathbf{w}^T\phi(x_i, y_i) - \mathbf{w}^T\phi(x_i, y) \geq \Delta(y_i, y) - \xi_i, \forall y \in \mathcal{Y}\backslash y_i,\ \ \xi_i \geq 0\end{gathered} \tag{10}$$

The prediction of a test example x is then given by $y^* = \text{argmax}_{y\in\mathcal{Y}}\, \mathbf{w}^{*T}\phi(x, y)$.

As the labels are structured, $|\mathcal{Y}|$ is often prohibitively large, making Eq. (10) intractable with standard SVM solvers. Iterative cutting-plane algorithms have been developed [18, 19], where the "most violated" constraints are identified by repeatedly solving the so-called separation oracle $\tilde{y}_i = \text{argmax}_{y\in\mathcal{Y}}\, \Delta(y_i, y) + \tilde{\mathbf{w}}^T\phi(x_i, y)$, and are added to the constraint set. These greedy algorithms admit polynomial training time, and are general in the sense that a large class of SOL problems (including sequence labelling) can be solved provided: 1) a joint feature mapping is defined, either explicitly as $\phi(x_i, y_i)$, or implicitly through a joint kernel function $J((x_i, y_i), (x_j, y_j)) =< \phi(x_i, y_i), \phi(x_j, y_j) >$; 2) a label loss function $\Delta(y_i, y)$ is specified; 3) an efficient algorithm for the separation oracle is available.

Now consider the event sequence labelling problem in court game annotation. Let (x_i, y_i) and (x_j, y_j) be two training examples with length (number of events, or number of tokens) o_i and o_j respectively. We implement first order Markovian assumption through a joint kernel function:

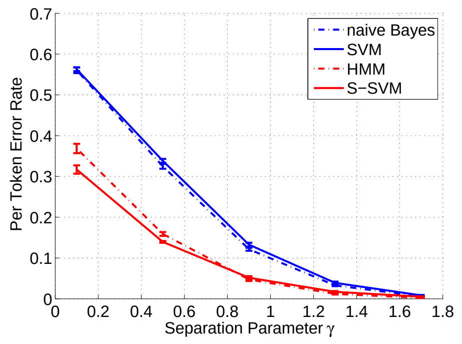
$$\begin{aligned} J((x_i, y_i), (x_j, y_j)) = & \sum_{s=2}^{o_i} \sum_{t=2}^{o_j} [\![y_i^{s-1} = y_j^{t-1}]\!] [\![y_i^s = y_j^t]\!] \\ & + \eta \sum_{s=1}^{o_i} \sum_{t=1}^{o_j} [\![y_i^s = y_j^t]\!] K(\mathbf{x}_i^s, \mathbf{x_j}^t) \end{aligned}$$

where $[\![\cdot]\!]$ is an indicator function, η is a kernel parameter controlling the trade-off between temporal correlation and observation, and $K(\mathbf{x}_i^s, \mathbf{x_j}^t)$ is a kernel defined for the observations [20, 21].

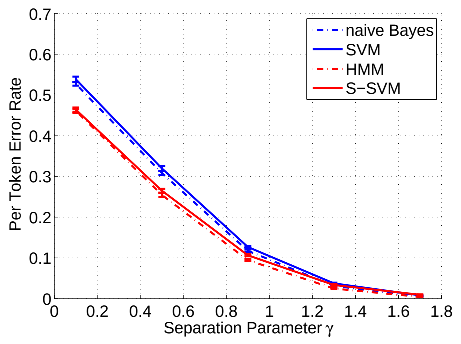
For the label loss function, we use the hamming loss between two competing labels: $\Delta(y_i, y) = \sum_{s=1}^{o_i} [\![y_i^s \neq y^s]\!]$. Finally, it is easy to show that the separation oracle for sequence labelling is the Viterbi decoding problem, for which efficient algorithms exist.

A simulation. Consider event sequences where the set of micro-labels is $\{serve, hit, bounce, net\}$. We employ two ways of simulating successive events. The first makes a first order Markovian assumption, while the second assumes no dependence between neighbouring tokens. For each token in each sequence a 10 dimensional observation is also simulated using one of four distributions corresponding to the four types of events. The separation of the distributions is controlled by varying their means through a parameter γ: the larger its value, the more separated. We consider two scenarios for observation distributions, namely, Gaussian and uniform.

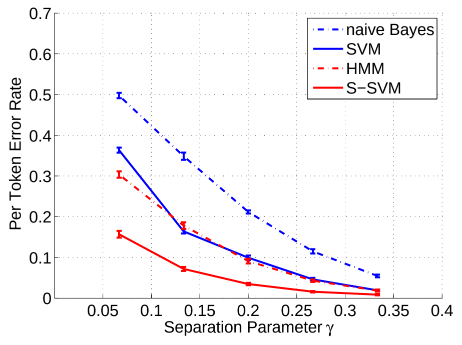
The combination of Markovian/non-Markovian and Gaussian/uniform results in a total of four scenarios. For each of them, 2000 sequences of micro-labels and associated observations are simulated. We use 1000 sequences for



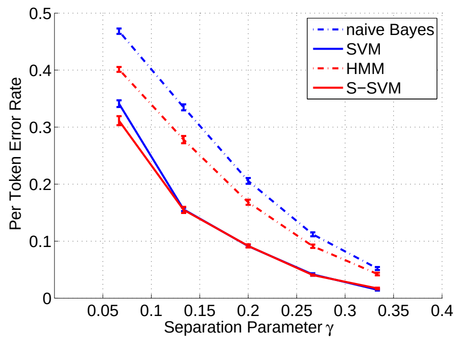
(a) Markovian + Gaussian



(b) Non-Markovian + Gaussian



(c) Markovian + Uniform



(d) Non-Markovian + Uniform

Figure 7: Mean and standard deviation of test error rates as functions of the separation parameter γ.

training, and the rest for testing. For the two generative methods, Gaussian distributions are assumed for the observations. The mean and standard deviation of the test errors of the four learning methods in the four simulated scenarios are shown in Fig. 7.

The results in Fig. 7 demonstrate that when there is indeed structure in the micro-labels, structured classification methods outperform non-structured ones; while when no structure is present, both methods perform similarly. On the other hand, when $P(x, y)$ is estimated poorly, the performance of a generative approach is severely degraded. In contrast, discriminative approaches do not make assumptions of $P(x, y)$, and as a result tend to be more robust [22]. Overall, the structured and discriminative S-SVM seems to be a safe choice: in all the scenarios considered, it produces either optimal or near optimal performance. We will test if this is still the case on real world data in the next section.

5. Experiments

In this section, we evaluate our proposed approach to tennis annotation on three real world tennis games, namely, the Australian Open 2003 women's singles final game, the Australian Open 2003 men's singles final game, and the Australian Open 2008 women's doubles final game. Statistics of the games are provided in Table 3. Note that to get ground truth detailed annotation of events is required, which is a laborious task. For each game, the leave-one (shot)-out error is computed. We compare the four learning techniques, and investigate the performance of different features. We consider three cases: using the ball trajectory based feature Eq. (5) alone; using the audio based feature Eq (7) alone, and combination of both features by early fusion, i.e., concatenation.

The results are summarised in Tables 4 to 6 . For all nine combinations of dataset and feature type, structured methods perform the best. Among them, HMM is the winner for two combinations, and S-SVM is the winner for the remaining seven. This clearly indicates that the micro-labels in real world games are structured, even with errors inevitably introduced during ball tracking and event detection. Overall, discriminative methods outper-

Table 3: Statistics of datasets.

	Aus03 Women Sing.	Aus03 Men Sing.	Aus08 Women Doub.
number of frames	100772	193384	355048
number of events	628	895	2325
number of play shots	71	90	163

Table 4: Leave-one-out per-token error rate: Australian 2003 Women Singles.

	naive Bayes	SVM	HMM	S-SVM
trajectory	22.27	10.27	10.74	**8.53**
audio	-	44.08	50.71	**33.49**
trajectory + audio	-	10.43	11.37	**7.74**

form generative ones: SVM is significantly better than naive Bayes in almost all cases, while S-SVM is better than HMM in seven, and is only marginally behind in the other two. This observation suggests that the features we use are not strictly Gaussian. On all the three datasets, the best performance is achieved with S-SVM, which matches our observation in the simulation. Note that for the 2003 Women Singles game the performance of naive Bayes is not reported for the audio based feature and for the trajectory + audio feature. This is because the one-dimensional audio feature has zero variance for one event class; as a result, naive Bayes cannot be performed for this dataset.

Comparing the visual and audio feature performances, we notice that on the Australian 2008 doubles game, where the audio feature has the lowest

Table 5: Leave-one-out per-token error rate: Australian 2003 Men Singles.

	naive Bayes	SVM	HMM	S-SVM
trajectory	30.28	15.75	16.42	**13.52**
audio	65.36	63.69	**63.35**	64.47
trajectory + audio	30.61	16.09	17.54	**13.74**

Table 6: Leave-one-out per-token error rate: Australian 2008 Women Doubles.

	naive Bayes	SVM	HMM	S-SVM
trajectory	20.13	12.51	13.58	**11.98**
audio	34.98	35.04	**27.64**	28.12
trajectory + audio	18.58	10.38	11.50	**9.80**

error among the three datasets, fusing the audio feature with the trajectory feature improves upon trajectory feature alone, for all the four learning methods. On the other two datasets however, including the audio feature does not always have a positive effect: on the Australian 2003 women's game, only S-SVM benefits from the fusion; while on the Australian 2003 men's game, the fusion has a small negative impact on all the learning methods.

In order to understand why the audio feature does not always help, we look more closely at its quality. We threshold the audio feature and use the binary result as a detector for the hit event, and show its performance in terms of precision and recall in Table 7. We also show in the same table the leave-one-out error rate of using the audio feature alone, and the improvement over the trajectory feature alone by fusion. The correlation is clear: when

Table 7: Quality of feature and its impact on performance. Learning method: S-SVM.

		Aus03 Women Singles	Aus03 Men Singles	Aus08 Women Doubles
hit detection from audio	precision	72.62	10.66	79.77
	recall	63.02	16.71	72.25
leave-one-out per-token error	audio	33.49	64.47	28.12
	improvement	0.79	-0.22	2.18

the audio quality is poor, as indicated by low precision and recall in hit event detection, the event classification error of the audio feature is high, and the improvement of fusion is also low (or even negative). By manual inspection, the very low quality of the audio feature on the Australian 2003 men's game is due to a serious acoustic mismatch between the training data and this set.

In Table 8 we show the change in the confusion matrix when the audio features are introduced: positive numbers on the diagonal and negative numbers off the diagonal indicate an improvement. On the dataset where the audio feature is not helpful (Australian 2003 men's single game, Table 8 left), there is little change in the confusion matrix. On the dataset where the audio feature helps (Australian 2008 women's double game, Table 8 right), we can see how it does: with the audio feature, ambiguities around hit events are reduced. This is expected, as the audio feature is defined as the confidence measure of the hit audio event. We tried using confidence measures associated with other audio events, but this did not yield any noticeable

Table 8: Change of confusion matrix due to fusion. Learning method: S-SVM. Left: Aus03 men single. Right: Aus08 women double.

	S	B	H	N	nu.
S	0	0	0	0	0
B	0	-1	-1	+1	+1
H	0	0	-3	0	+3
N	0	0	0	0	0
nu.	-1	-1	0	0	+2

	S	B	H	N	nu.
S	0	0	-1	0	+1
B	-1	-2	-2	+1	+4
H	-3	-7	+14	0	-4
N	0	-3	-1	+5	-1
nu.	-1	+1	-26	+2	+24

improvement.

In the future, we plan to incorporate player action recognition and natural language processing (NLP) on the commentary transcripts to help event classification, and the annotation in general. We would also like to investigate methods that can exploit audio information more effectively. Finally, the event sequences produced in this work are still to be integrated with a tennis court detection module we have developed (cf. Fig. 1), to produce higher level annotation, e.g. the scores.

6. Conclusions

In this paper we have presented a solution to the challenging problem of automatic annotation of tennis game using broadcast tennis videos. The output of our system is key events with locations in the row-column-time space and labels, which can potentially lead to scores. To the best of our knowledge, our work is the only one that can annotate at such a detailed level. This is achieved by integrating computer vision, audio processing, and machine learning. For each of the disciplines involved we have designed our approach carefully so as to optimise the overall performance. The proposed method was evaluated on three real world tennis videos with positive results. Discussions on the interplay between vision, audio, and learning, and on future work plan were also provided.

Acknowledgement

This work has been supported by EPSRC/DSTL project EP/K014307/1 and EU Chist-Era EPSRC EP/K01904X/1 Visual Sense project.

References

[1] E. Kijak, G. Gravier, P. Gros, L. Oisel, F. Bimbot, HMM Based Structuring of Tennis Videos Using Visual and Audio Cues, in: IEEE Internatinal Conference on Multimedia and Expo, vol. 3, 309–312, 2003.

[2] I. Kolonias, W. Christmas, J. Kittler, Tracking the Evolution of a Tennis Match Using Hidden Markov Models, in: Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition, vol. 3138, 1078–1086, 2004.

[3] Y. Huang, C. Chiou, F. Sandnes, An Intelligent Strategy for the Automatic Detection of Highlights in Tennis Video Recordings, Expert Systems with Applications 36 (2009) 9907–9918.

[4] The Hawkeye Tennis System, http://www.hawkeyeinnovations.co.uk, 2014.

[5] X. Yu, C. Sim, J. R. Wang, L. Cheong, A Trajectory-based Ball Detection and Tracking Algorithm in Broadcast Tennis Video, in: ICIP, vol. 2, 1049–1052, 2004.

[6] B. Ekinci, M. Gokmen, A Ball Tracking System for Offline Tennis Videos, in: International Conference on Visualization, Imaging and Simulation, 2008.

[7] G. Zhu, C. Xu, Q. Huang, W. Gao, Action Recognition in Broadcast Tennis Video, in: International Conference on Pattern Recognition, 2006.

[8] F. Yan, J. Kittler, K. Mikolajczyk, D. Windridge, Automatic Annotation of Court Games with Structured Output Learning, in: International Conference on Pattern Recognition, 2012.

[9] W. Christmas, A. Kostin, F. Yan, I. Kolonias, J. Kittler, A System for the Automatic Annotation of Tennis Matches, in: Fourth International Workshop on Content-Based Multimedia Indexing, 2005.

[10] F. Yan, W. Christams, J. Kittler, Layered Data Association Using Graph-Theoretic Formulation with Application to Tennis Ball Tracking in Monocular Sequences, PAMI 30(10) (2008) 1814–1830.

[11] E. Kijak, G. Gravier, L. Oisel, P. Gros, Audiovisual Integration for Tennis Broadcast Structuring, in: International Workshop on Content-Based Multimedia Indexing, 2003.

[12] F. Coldefy, P. Bouthemy, M. Betser, G. Gravier, Tennis Video Abstraction from Audio and Visual Cues, in: IEEE International Workshop on Multimedia Signal Processing, 2004.

[13] G. Zhu, Q. Huang, C. Xu, L. Xing, Human Behavior Analysis for Highlight Ranking in Broadcast Racket Sports Video, IEEE Transactions On Multimedia 9(6) (2007) 1167–1182.

[14] J. Lai, C. Chen, C. Kao, S. Chien, Tennis Video 2.0: A New Presentation of Sports Videos with Content Separation and Rendering, Journal of Visual Communication and Image Representation 22 (2011) 271–283.

[15] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, second edn., 2004.

[16] J. Kittler, W. Christmas, A. Kostin, F. Yan, I. Kolonias, D. Windridge, A Memory Architecture and Contextual Reasoning Framework for Cognitive Vision, in: 14th Scandinavian Conference on Image Analysis, 2005.

[17] Q. Huang, S. Cox, F. Yan, T. Campos, D. Windridge, J. Kittler, W. Christmas, Improved Detection of Ball Hit Events in a Tennis Game Using Multimodal Information, in: International Conference on Auditory Visual Speech Processing, 2011.

[18] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large Margin Methods for Structured and Interdependent Output Variables, JMLR 6 (2005) 1453–1484.

[19] T. Joachims, T. Finley, C. Yu, Cutting-Plane Training of Structural SVMs, Machine Learning 77 (2009) 27–59.

[20] Y. Altun, Discriminative Methods for Label Sequence Learning, PhD Thesis, Department of Computer Science, Brown University, USA, 2005.

[21] B. Taskar, C. Guestrin, D. Koller, Max-Margin Markov Networks, in: NIPS, 2003.

[22] A. Ng, M. Jordan, On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes, in: NIPS, 2002.