# A memory architecture and contextual reasoning framework for cognitive vision

J. Kittler, W.J. Christmas, A. Kostin, F. Yan, I. Kolonias and D. Windridge *

Centre for Vision, Speech and Signal Processing
University of Surrey , Guildford, GU2 7XH, UK
w.christmas@surrey.ac.uk

**Abstract.** One of the key requirements for a cognitive vision system to support reasoning is the possession of an effective mechanism to exploit context both for scene interpretation and for action planning. Context can be used effectively provided the system is endowed with a conducive memory architecture that supports contextual reasoning at all levels of processing, as well as a contextual reasoning framework. In this paper we describe a unified apparatus for reasoning using context, cast in a Bayesian reasoning framework. We also describe a modular memory architecture developed as part of the VAMPIRE * vision system which allows the system to store raw video data at the lowest level and its semantic annotation of monotonically increasing abstraction at the higher levels. By way of illustration, we use as an application for the memory system the automatic annotation of a tennis match.

## 1 Introduction

Over the last decade, the subject of visual perception has made considerable progress. Novel learning algorithms (Support Vector Machines, AdaBoost, Multiple Classifier Systems) have been developed and combined with geometric invariance, picture formation and noise models to enhance the capability of machine perception systems. These have now reached the stage of maturity that makes them deployable in constrained or semi-constrained environments to perform tightly specified visual tasks.

However, in spite of the progress, the state of the art systems which rely heavily on machine learning lack many features characteristic of human cognition, in particular the ability to understand and reason about the perceived environment, learn new concepts, develop through interaction with other systems and users, define its own perceptual goals as well as to control the perception process, and use the perceived information to support action.

One of the key requirements for a cognitive vision system to support reasoning is to possess an effective mechanism to exploit context both for scene interpretation and for action planning. Context is conveyed by the sensory data

* This work was funded by EC projects IST-34401 "VAMPIRE", IST-004176 "COSPAL" and IST-507752 "MUSCLE"

itself, but also by the model of the imaged scene and the a priori model of the universe in which the perception system operates. By context we understand relationships between the (intrinsic and extrinsic) properties of objects appearing in the scene. Context is omnipresent and can aid vision processing at all levels. At the lowest level it is conveyed by pixels neighbouring in space and time. At high levels it pertains to objects or sets of objects that define more abstract semantic entities in a particular application domain.

Context can be used effectively provided the system is endowed with a conducive memory architecture that supports contextual reasoning at all levels of processing, as well as a contextual reasoning framework. In this paper we describe a memory architecture developed as part of the VAMPIRE vision system which allows the system to store raw video data at the lowest level and its semantic annotation of monotonically increasing abstraction at the higher levels. The memory mechanisms include forgetting processes with the rate of forgetting being inversely related to the level of interpretation. Thus the memory of the raw input data decays rapidly where as the high levels retain the ability to provide symbolic description of the scene over relatively long term. The architecture supports forward and feedback interaction between levels.

We also describe a unified apparatus for reasoning in context. It is cast in the Bayesian framework of evidential reasoning. The use of both the memory structure and the reasoning engine is illustrated on the problem of interpreting tennis videos. We show that the contextual reasoning can be applied to the problem of foreground/background separation at the lowest level of the system, through tennis ball and player tracking, to high level reasoning about score points.

The paper is organised as follows. In the next section we develop the unified Bayesian framework for contextual reasoning. In Section 3 we summarise the proposed memory architecture. The following section (Section 4) describes the individual processing modules. Section 5 describes some experiments in which we ran the complete system on some of the material from the women's final of the 2003 Australian Open Tennis Tournament. In the final section we review the progress made on the annotation system, highlighting some of the remaining problems, and outlining the future direction of this work.

## 2 Theoretical Framework

In order to develop a general framework for interpreting video data we need to introduce an appropriate mathematical notation. Let $v_i^t,\ i = 1, \ldots, N_t$ be a set of video objects to be interpreted at time $t$. The nature of these objects will very much depend on the interpretation task and the level of video content representation. At the lowest level the objects may be *pixels*, whereas at higher levels they may be regions, groups of regions, or visual events.

The inherent spatial relation of these objects is best represented in terms of an attributed relational graph $\mathbf{G^t} = (\mathbf{V^t}, \mathbf{E^t}, \mathbf{X^t}, \mathbf{B^t})$ where $\mathbf{V}^t$ is a set of vertices (nodes) constituted by objects $v_i^t$ and $\mathbf{E}^t$ is the set edges $e_{ij}^t, i = 1, \ldots, N_t,\ j \epsilon \mathfrak{N}_i$

connecting object $v_i^t$ with $v_j^t$ neighbouring to $v_i^t$. $\mathfrak{N}_i$ defines the index set of neighbours to node $i$ within a particular neighbourhood system. For instance, at the lowest level, the neighbourhood system could be a 2D lattice where as at higher levels the neighbourhood system could be a general, fully connected graph. $\mathbf{X}^t$ and $\mathbf{B}^t$ denote unary and binary relation information characterising the nodes of the graph, i.e.

$$\begin{array}{l} \mathbf{X}^t = \{\mathbf{x}_i | i = 1, \ldots, N_t\} \\ \mathbf{B}^t = \{\mathbf{b}_{ij} | i = 1, \ldots, N_t, \ j \epsilon \mathfrak{N}_i\} \end{array} \tag{1}$$

where $\mathbf{x}_i$ is a vector of unary attributes relating to node $i$ and $\mathbf{b}_{ij}$ is a vector of binary relations between objects $i$ and $j$.

Each object is assumed to have a unique identity, determined by its intrinsic properties (shape, colour, texture) and extrinsic properties ( pose, position, motion) that are the basis for its symbolic grounding. The measurement of these properties may be subject to sensor transfer function and imaging transformation. Let us denote the identity of object $v_i^t$ by $\theta_i^t$. Then the problem of video interpretation can be formulated as one of finding the most probable labelling $\theta_i^t = \omega_{\theta_i^t}$ of objects $v_i^t, \ i = 1, \ldots, N_t$, given the measurement information conveyed by the attributed relational graphs at all the time frames up to time $t$ as well as the interpretation of all the objects in the previous frames. Adopting a shorthand notation $\Theta^k = (\theta_1^k, \ldots, \theta_{N_k}^k)$ as a label set and $\Omega^k = (\omega_{\theta_1^k}, \ldots, \omega_{\theta_{N_k}^k})$ as the set of specific identities assumed by labels in $\Theta^k$, the interpretation problem can be stated

$$\begin{array}{l} assign \ v_i^t \rightarrow \omega_{\theta_i^t}, \forall i \quad if \\ \qquad P(\Theta^t = \Omega_{\Theta^t} | G^t, \ldots, G^1, \Theta^{t-1}, \ldots, \Theta^1) = \\ \qquad = \max_\Omega P(\Theta^t = \Omega | G^t, \ldots, G^1, \Theta^{t-1}, \ldots, \Theta^1) \end{array} \tag{2}$$

Note that in equation (2) the a posteriori probability can be expressed as

$$\begin{array}{l} P(\Theta^t = \Omega | \ G^t, \ldots, G^1, \Theta^{t-1}, \ldots, \Theta^1) = \\ \qquad = \frac{p(G^t, \ldots, G^1, \Theta^{t-1}, \ldots, \Theta^1 | \Theta^t = \Omega) P(\Theta^t = \Omega)}{\sum_\Lambda p(G^t, \ldots, G^1, \Theta^t = \Lambda, \Theta^{t-1}, \ldots, \Theta^1)} \end{array} \tag{3}$$

This can be further rearranged as follows:

$$\begin{array}{l} P(\Theta^t = \Omega | \ G^t, \ldots, G^1, \Theta^{t-1}, \ldots, \Theta^1) = \\ \qquad = \frac{p(G^t, \ldots, G^1 | \Theta^t = \Omega, \Theta^{t-1}, \ldots, \Theta^1) P(\Theta^t = \Omega, \Theta^{t-1}, \ldots, \Theta^1)}{\sum_\Lambda p(G^t, \ldots, G^1 | \Theta^t = \Lambda, \Theta^{t-1}, \ldots, \Theta^1) P(\Theta^t = \Omega, \Theta^{t-1}, \ldots, \Theta^1)} \end{array} \tag{4}$$

As the denominator in (4) is independent of labelling $\Omega$ it will not affect the decision making process and can be ignored. The contextual interpretation of the objects at time instant $t$ is a function of the likelihood of observing measurements $G^t, G^{t-1}, \ldots, G^1$ given jointly the hypothesis $\Omega$ as well as the interpretation of the objects in the past frames, and the prior probability of the interpretation up to time $t$. Under the assumption that both the measurement and label processes are Markovian, i.e. the past history is captured by the measurements and labels

at the previous time frame, we can simplify (4) as

$$P(\Theta^t = \Omega |\, G^t, \ldots, G^1, \Theta^{t-1}, \ldots, \Theta^1) =$$

$$= \frac{p(G^t|G^{t-1}, \Theta^t=\Omega, \Theta^{t-1}) P(\Theta^t=\Omega|\Theta^{t-1}) p(G^1|\Theta^1) P(\Theta^1)}{\sum_\Lambda p(G^t, \ldots, G^1|\Theta^t=\Lambda, \Theta^{t-1}, \ldots, \Theta^1) P(\Theta^t=\Omega, \Theta^{t-1}, \ldots, \Theta^1)} \times \tag{5}$$

$$\times \prod_{k=1}^{t-2} p(G^{t-k}|G^{t-k-1}, \Theta^{t-k}, \Theta^{t-k-1}) P(\Theta^{t-k}|\Theta^{t-k-1})$$

or more intuitively as

$$P(\Theta^t = \Omega |\, G^t, \ldots, G^1, \Theta^{t-1}, \ldots, \Theta^1) =$$

$$= \frac{p(G^t|G^{t-1}, \Theta^t=\Omega, \Theta^{t-1}) P(\Theta^t=\Omega|\Theta^{t-1}) P(\Theta^{t-1}|G^{t-1}, \ldots, G^1, \Theta^{t-2}, \ldots, \Theta^1)}{norm} \tag{6}$$

which shows that the interpretation of video objects at time $t$ is a function of the interpretation deduced at time $t-1$, the probability of transition from state $\Theta^{t-1}$ to $\Theta^t$ and the likelihood of observing measurements $G^t$ given measurements $G^{t-1}$ and labels. The latter is an objective function of attributed relational graph matching which realises spatial contextual reasoning under temporal consistency constraints. If there is no temporal context (for example when performing camera calibration in our system) the problem reduces to a standard graph matching one [3, 6, 8]. Conversely if the only context is temporal, we can employ the standard techniques of Kalman or particle filters [4], or hidden Markov models [11].

The spatio-temporal Bayesian reasoning embodied by (6) provides a unifying framework for combining evidence in a wide range of video interpretation tasks at various levels of video representation. In our tennis video analysis test bed these include foreground / background separation, tennis ball tracking, player tracking, tennis court detection, event detection, and high level analysis. For each task the framework has been further developed to take into account the the pertinent spatio-temporal constraints. In Section 4 we will give a more detailed account of a few of these tasks to illustrate the generality of the framework. First, the adopted memory architecture supporting the reasoning process will be described in the next section.

## 3 The video memory system

Working memory is effectively a capacity-limited audio-visual buffer that has been empirically demonstrated to exist in human subjects via the techniques of cognitive psychology. In visual terms its capacity is taken to be analogous to the extent and resolution of a spatial buffer, and is hence considered the location of our 'mental imagery' processing [7]. However, according to Just [5], it is possible utilise this buffer for visual processing as well as representation. Thus, it is possible in working memory to envisage sequences of geometric transformations of relatively simple images as well as relatively complex but static images, as the current cognitive task demands. Working memory is hence a finite computational resource that is allocated on demand by a central executive.

In typical cognitive task (one such as sentence comprehension that is balanced equally between representation and relationship) short term working memory has an effective capacity of around seven distinct 'chunks' of data [9], a chunk being a pattern of audio-visual data that has been committed to long term memory. The discrete chunks in working memory are stored only very briefly, but may be recalled by the central executive instantly and in full detail (that is, with all of their associated attributes). Storage in the long-term memory, on the other hand, is primarily associative, relating very large numbers of differing items to one another in terms of their co-occurrences, rather than via their inherent attributes. Recall from the long term memory is thus a slow process, and (lacking immediate access to the attribute data) completely reliant on a sufficiency of retrieval cues related by association to the item currently under consideration in order to be appropriately recalled.

The discrete patterns represented in long term memory are hence high level abstractions of the underlying audio-visual sensory representations existing at the level of the short-term working memory, and are originally allocated on the basis of the number of times that that particular set of attributes has occurred within the working memory [1]. There is hence an inverted relationship between memory retention and interpretative level amongst human subjects, with the lowest, least generalised level of sensory memory subject to the fastest decay times (in the short-term memory), and the highest-levels of associative and relational data retained for the longest times (in the long-term memory).

In the system we have constructed, the memory system is in two separate parts: short-term and long-term, together with a graphical browser to access the memory contents. The short-term memory analyses the video, in a bottom-up fashion, making available relevant results to the long-term memory, and "forgetting" the rest. The long-term memory stores only the information that might be useful in the longer term. An analogy with human memory might be that the short-term memory is active when the system is watching the tennis match, working out what is going on, and remembering the exciting bits.

### 3.1 The short-term memory

The short-term memory provides the infrastructure to run the particular set of modules selected for the application (although the short-term memory system itself is designed to be independent of the application). Thus it includes:

- self-assembly of the system
- the means of starting the modules as individual threads
- communication between the modules
- storage for the module data
- the re-use of previously computed partial results
- forgetting data that is no longer wanted

The system is run by launching a target module (or modules); this module then launches those modules needed to generate its input data, in a recursive fashion, until all of the modules needed for the application have been assembled.

### 3.2 The long-term memory

The long-term memory becomes active once the short-term memory has generated annotation, recalling exciting moments in the game, and remembering the general outcome, such as: who won, how close it was, how many sets, maybe who won each set. The long-term memory is thus relatively static once it is populated: it is the short-term memory that is actively and progressively refining the video data to extract meaning from it. Unlike the short-term memory design, it is (currently) specific to the application.

## 4 Processing modules for the short-term memory

The system is intended for use with low-quality, off-air video from a single camera (unlike for example [10]). The ball tracking in particular is a challenging task under these conditions: the movement of the tennis ball is so fast that sometimes it is blurred into the background, and is also subject to temporary occlusion and sudden change of motion direction. Two example frames are shown in Fig. 1. In Fig. 1(a), the tennis ball is over the player's head, with a size of only about five pixels. In Fig. 1(b), the tennis ball velocity is very high, and is almost completely blurred into the background.

(a) Far player serving (b) Fast moving tennis ball

**Fig. 1.** Frames in which the ball is difficult to track

The complete set of modules that make up the tennis annotation system is shown in Fig. 2, which also shows the data flow between the modules. Most of

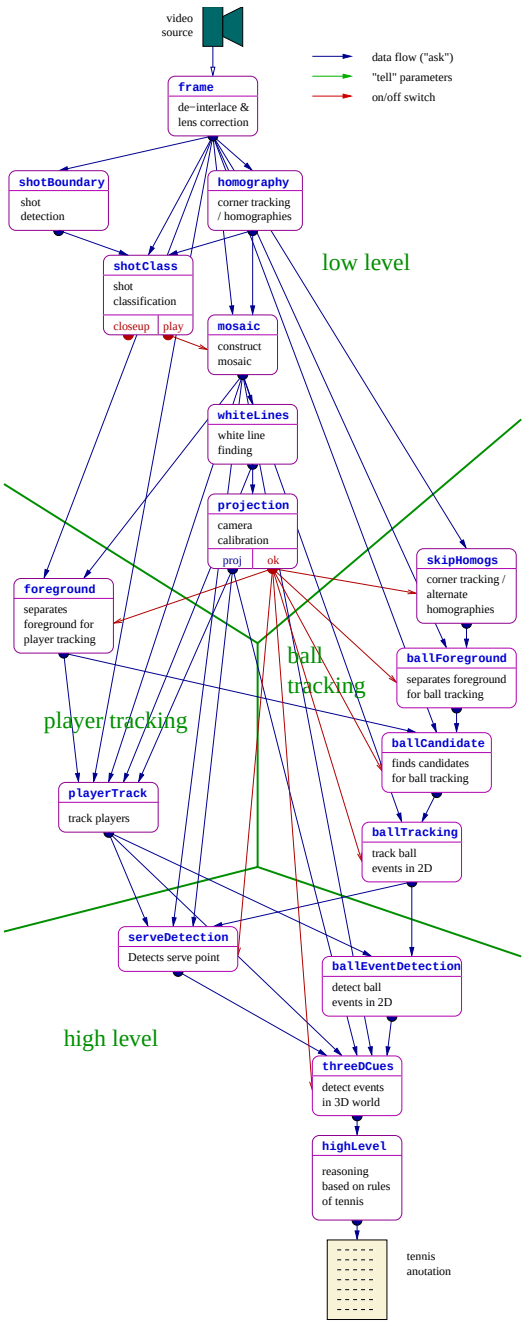



**Fig. 2.** Module set and data flow for tennis annotation

the low-level modules are run on the whole video. The remaining modules are only run on play shots.

### 4.1 Low-level modules

Many of these modules do not require state-of-the-art algorithms for this application. Accordingly they were mainly implemented using simple, fast, established algorithms (with exception of the mosaic module).

**Module 'frame'** The 'frame' module reads a stream of video frames, deinterlaces them into a stream of fields and applies a preset amount of lens distortion correction.

**Module 'homography'** The camera position on the court is assumed to be fixed, so that the global transformation between consecutive pairs of fields can be assumed to be a homography. 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 to improve the homography.

**Module 'shotBoundary'** This module uses the colour histogram intersection between adjacent fields to detect shot boundaries.

**Module 'shotClass'** A linear classifier is used to classify the shots into "play" and "non-play" using a combination of colour histogram mode and corner point continuity. Some shots are incorrectly classified (for our purposes) as "play", such as replays. However these false positives are generally eliminated later on by the module 'projection', which rejects the shot if it is unable to find the court markings.

**Module 'mosaic'** A mosaic is generated for each "play" shot as follows:

- The inter-field homographies are used to approximately warp each field into the mosaic coordinate system.
- The warped field is re-registered with the current mosaic to improve the homography.
- The warped fields are subsampled — typically 1 in 10 fields are retained for the mosaic.
- The mosaic is constructed by applying a median filter to each pixel of the set of these remaining warped fields.

**Module 'whiteLines'** The white lines are located using a combination of edge detector and Hough transform.

**Module 'projection'** At present, some basic assumptions are made about the camera position in order to label the white lines. A court model is then used to set up a set of projective linear equations in order to solve for the camera homography. If the module fails, the shot is relabelled as "non-play".

### 4.2 Player tracking modules

**Module 'foreground'** The foreground is determined for each field by subtracting the warped mosaic from the field, low-pass filtering the difference, and applying a threshold to the filtered difference image. The result is then filtered to reduce the noise.

**Module 'playerTrack'** Firstly an attempt is made to locate the players from the foreground images from the 'foreground' module. This sometimes fails, e.g. because one of the players in a short shot does not move enough to be removed from the mosaic image. If this is the case, a second algorithm is invoked, in which the shot is segmented into regions, and the dominant colour established. Candidate regions are identified that are (a) significantly different form the dominant colour and (b) in a plausible position within the image.

### 4.3 Ball tracking modules

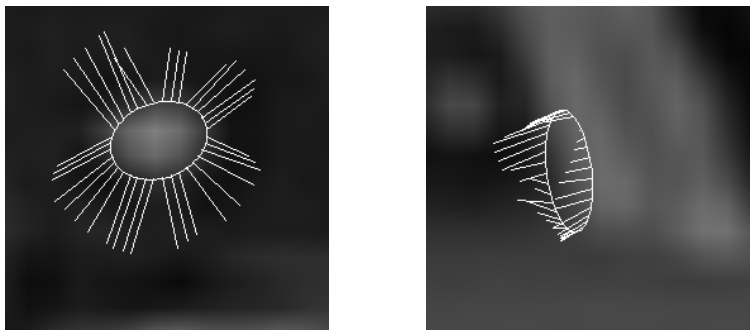
**Module 'skipHomogs'** Because of the spatial aliasing present in the fields, a second homography module was implemented for the ball tracking branch, in order to generate more precise foreground separation. This module operates in a broadly similar fashion to the 'homography' module. The difference is that the homographies are computed between corresponding fields of consecutive frames: i.e. field $n$ is compared with field $n+2$. Thus two separate corner trackers are needed, for the odd and even fields.

**Module 'ballForeground'** For the $i^{th}$ field, six fields, with sequence numbers $i-8, i-6, i-4, i+4, i+6, i+8$, are chosen as reference fields. Each motion-compensated reference field is subtracted from the $i^{th}$ field. The subtraction results are then thresholded to get 6 foreground images. A pixel is classified as a foreground pixel only if it appears as a foreground pixel on all the six foreground images. A morphological opening operation is then applied, to further remove noise.

**Module 'ballCandidates'** The foreground image may contain blobs from the rackets or the players, as well as from the true tennis ball. In a single frame, the ball is distinguished by its colour and shape. However we concluded that, due to the relatively poor colour rendering in off-air video, colour was of little practical use in classifying the blobs. On the other hand, the ball shape is consistently oval (circular if the ball is stationary).

The blob shape is analysed as follows. An ellipse is fitted to the blob boundary. The image gradient on the ellipse boundary is measured. If the blob is a ball, the gradient is assumed to be normal to the ellipse and positive in the direction of the ellipse centre (Fig. 3). The blobs are then classified (using the AdaBoost method) on this basis.

**Module 'ballTracking'** The proposed tennis ball tracking algorithm is based on a Kalman filter and data association technique [2]. The Kalman filter works well when the linear assumption of tennis ball motion is satisfied, that is, except when the tennis ball bounces or is hit. The position, velocity and acceleration of



(a) tennis ball (b) other blob

**Fig. 3.** Gradient vectors superimposed on candidate blobs

the tennis ball in both $x$ and $y$ directions are modelled in the filter state vector. To enhance the basic filter, the following measures are taken:

- Because the ball motion experiences a discontinuity when it bounces or is hit which is not modelled by the Kalman filter, the method permits the generation of multiple hypotheses at each new field. A "lifetime" parameter, which represents the number of "no observation fields" a track can tolerate before it is discarded, is defined for each track. The life time is associated with the cumulative likelihood of the track: the higher the cumulative likelihood, the longer the life time. Consequently, a track that is more likely to be the true trajectory has a better chance of surviving the failure of the object detection. Finally, at each observation the track that has greatest cumulative likelihood is selected (magenta track in Fig 4).
- Since the track is generally reliable before a motion discontinuity, but is less reliable afterwards, the tracking is performed in both backward and forward directions. The two tracks are merged field-by-field, on the basis of maximum likelihood.

### 4.4 The high-level modules

These modules combine the results from the ball and player tracking to generate an analysis of the evolution of each play shot.

**Module 'ballEventDetection'** The progress of the tennis match is described by key events such as: the tennis ball being hit, bouncing on the court or hitting the net. Such events can be spotted by detecting the motion discontinuities of the tennis ball. A point on the trajectory is marked as an event when the change in both orientation and motion magnitude exceed specific thresholds.

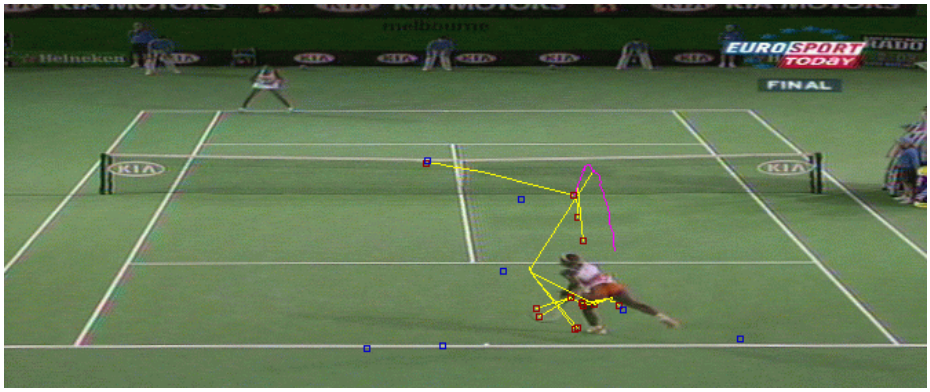



**Fig. 4.** Making multiple hypotheses

An example of the final tracking result and event detection result is shown in Fig. 5. In this example there are no false positive events; however 3 events (1 bounce, 2 hits) are not detected. It is also left to higher level modules to recover from wrong decisions made here.

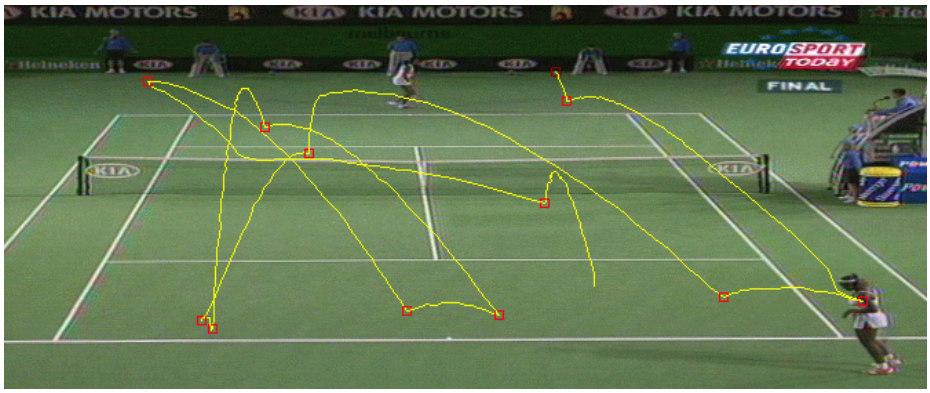



**Fig. 5.** Final tracking result (with event detection)

**Module 'serveDetection'** The system performs these three operations on each player:

- process player tracking data to see whether any of the players is located in a possible serving position, and create a contour of each player,
- detect whether any of the players has the body pose we would normally associate with a serve hit,

– verify that the tennis ball is directly above the player.

The process is repeated for each field from the beginning of the shot, and terminates if, for *any* of the players *all* of the above hold. At this point a serve is deemed to have occurred.

**Module '3DCues'** The events from the preceding modules are reinterpreted in 3D space:

– Hit / bounce discrimination is performed on all events recognised by the ball tracking module. If a player is close to the ball and the ball is located where a hit can reasonably be made, the event is labelled as a hit; otherwise it is labelled as a bounce.
– The court coordinates of the players and of ball events are projected onto the court surface model. We use the position of the players' feet, and assume that when the ball is hit it is roughly above the player.
– The events are labelled according to the region of the court in which they occurred.
– To avoid multiple instances of the same event in close succession, no event labels are permitted within 3 frames of each other.

**Module 'highLevel'** The rules of the game of tennis provide us with a good guideline as to what events we will have to be capable of tracking efficiently, so as to follow the evolution of a tennis match properly. Such events would include:

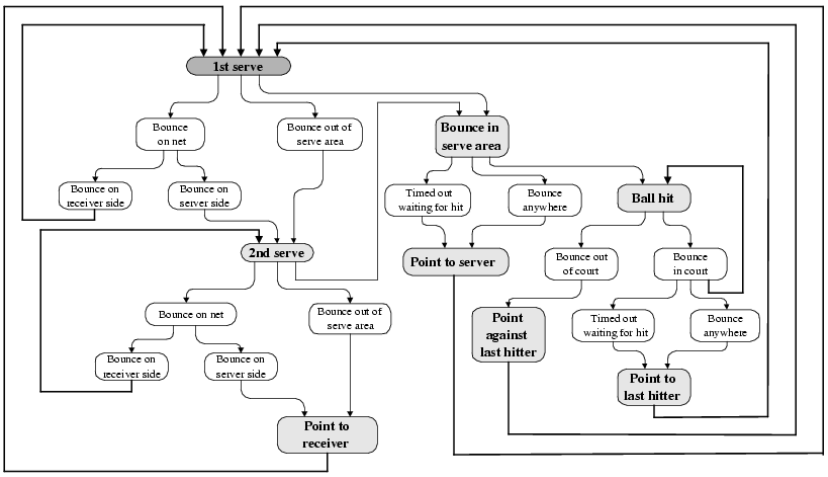
– the tennis ball being hit by the players
– the ball bouncing on the court
– the players' positions and shapes (that is, body poses)

These events are used to perform reasoning about events at a higher level, like awarding a play outcome. The model for the evolution and award of a point in a tennis match is illustrated in Fig. 6(a).

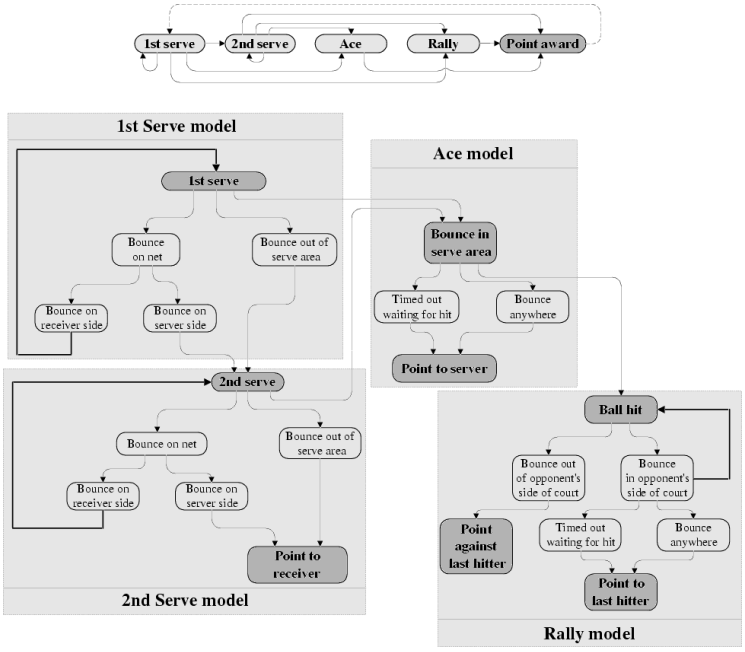
As we can see, the model contains a number of loops; the state transitions drawn with **bold** lines indicate where these loops close. In order to simplify the design, we propose to replace the original scene evolution model with a set of sub-models, each one illustrating a certain scenario of the match evolution. This set of sub-graphs is illustrated in Figure 6(b).

As we can see from the set of sub-models, we have opted for a more 'perceptual' way of selecting the set of event chains that will form the model. Moreover, choosing to decompose the initial graph down into sub-graphs and train each subgraph separately will be beneficial in other ways:

– Probabilistic reasoning tools (like HMMs) will enable correction of wrongly detected elementary events within the game, so a point can be correctly awarded even if not all events are accurately detected.
– Since the models are simpler, we can use a smaller training set.
– In some cases, we can use prior knowledge to speed up the training process. For example, statistics available for events occurring in a tennis match helps us get a very good initial estimate for the HMM parameters without the need for an explicit training process.




(a) Model for awarding a point in a tennis match




(b) Switching model and its set of sub-models

**Fig. 6.** State models for a tennis point

– It will be easier to determine which sub-models need to be improved to improve the whole system.

Since the system is dealing with ambiguous and noisy data, we are bound to encounter event detection errors in the process. Hence, another issue is that the the reasoning engine should be robust to false input events. To address this, the system used an HMM with a look-ahead decision mechanism, thereby allowing us to take into account events that occur *after* the current one. The length of the look-ahead window has been limited to one event, thus allowing us to correct isolated errors. There are two reasons for this choice:

– If the length of the look-ahead window is too large, short shots with errors in them may be wrongly interpreted. That can happen as, at the end of the chain, the Viterbi algorithm will not have enough evidence to correct errors.
– Of the events used as input to the HMM, the ones most susceptible to errors are the ball bounces. However, since only one bounce can occur between two successive hits if the ball is still in play, detecting a player hit (*not* a serve) automatically removes the chance of a point being awarded, even if the bounce point is wrongly detected (or not detected at all).

## 5 Experiments on the shot-level tennis annotation system

The scheme described above has been tested on approximately half an hour's play from the Women's Final of the 2003 Australian Tennis Open. The sequence included a total of 45 shots that were "play" shots, all of which were correctly recognised as such.

**Ball events** To measure the performance of the ball tracking modules, recall and precision of the event detection are used. Recall is defined as the ratio of true positives in detected events to the ground truth events; the precision is defined as the ratio of true positives in detected events to all detected events. Since an event is a motion discontinuity point in the trajectory, it corresponds to a point in X-Y-Time 3D space. A detected event is regarded as correct when it is within $\sqrt{3}$ pixels and 1 field of the ground truth position.

The experiments were carried out on 45 manually classified play shots, which contain 320 events. The results are as follows:

| Ground truth events | Precision | Recall | Performance |
|---|---|---|---|
| 320 | 0.85 | 0.81 | 0.65 |

The main cause of error is that is is sometimes difficult to detect that the ball has bounced — i.e. the deviation in the trajectory is very small. We can increase the sensitivity to such events, but at the cost of introducing false positives.

**Play events** If a shot contains an entire play sequence, there are 5 possible correct outcomes: no play; faulty serve by either player; point awarded to either player. The model also contains other possible situations, such as a good serve, but where the shot terminated before the play was complete. Out of 45 play shots, 18 were awarded correct outcomes. The causes of the erroneous shot outcomes can be roughly broken down as follows:

| Type of error | no. of affected shots |
|---|---|
| High-level reasoning engine | 14 |
| Ball event detection | 11 |
| Projecting events into 3D | 3 |
| Player tracking | 1 |

**Table 1.** Error summary

These results reflect the current state of development of the system. A more detailed analysis indicated a number of problems:

- Since we do not have enough data to train the HMM parameters in the high-level reasoning, the model had to be constructed by hand, based on the author's experience as a tennis enthusiast. The model is complex, and could undoubtedly be improved.
- It can be difficult to identify ball bounce events — sometimes the deflection of the ball *in the image plane* is barely perceptible.
- The system relies on the success of the serve detection algorithm in order to start processing a shot. If it fails, the sequence is not analysed further.
- If a shot contains more than one serve, the system only picks up the last one.
- Currently the reasoning engine gets re-initialised for each shot, thus eliminating any earlier information that could help us recover from errors encountered on the current shot.

## 6 Conclusions

We have created an integrated system that enables us to analyse tennis video material up to the level of a single shot. The system can also store partial results for subsequent experiments, thus avoiding repeated running of the time-consuming lower-level modules. The above results indicate that there is much to be done to improve the accuracy of the system, but they reflect the fact that the results come from one of the first experiments to be performed on the complete shot annotation system. Now that we have an integrated system, it is easier to identify the weaknesses, such as:

- Ball event detection: ball events can be hard to detect, even by eye. Also at present there is no explicit model of the ball events within the filter.

– Currently shots are analysed individually: combining information over wider time scales than a single shot, using higher-level models of the tennis rules, should improve the accuracy.
– The HMM interpreting the play rules in the 'highLevel' module is currently using as input "hard" decisions made in other modules. Using "soft" decisions, or combining decisions with confidence information, should improve matters.

However we feel that the basic approach is sound, and we emphasise that the results are preliminary. The memory model works reliably, and is readily adaptable to a wide range of cognitive tasks that require analysis at a number of different semantic levels.

## References


1. J.R. Anderson. *The architecture of cognition*. Harvard University Press, 1983.
2. Y. Bar-Shalom and T. E. Forman. *Tracking and Data Association*. Academic Press INC, 1988.
3. W.J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764, August 1995.
4. M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal on Computer Vision*, 1998.
5. M.A. Just, P.A. Carpenter, and D.D. Hemphill. Constraints on processing capacity: Architectural or implementational. In D.M. Steier and Mitchell T.M, editors, *Mind matters: A tribute to Allen Newell*. Erlbaum, 1996.
6. A. Kostin, J. Kittler, and W.J. Christmas. Object recognition by symmetrised graph matching using relaxation labelling with an inhibitory mechanism. *Pattern Recognition Letters*, 26(3):381–393, 2005. Special issue in memory of Professor Azriel Rosenfeld.
7. R.H. Logie. *Visuo-spatial working memory*. Lawrence Erlbaum Associates, 1995.
8. David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.
9. G. A Miller. The magical number seven, plus or minus two: Some limits of our capacity for processing information. *Psychological Review*, 63:81–97, 1956.
10. C. Harris N. Owens and C. Stennet. Hawk-Eye Tennis System. In *Proc. VIE 2003 – Intl. Conf. on Visual Information Engineering*, pages 182–185, July 2003.
11. L. R. Rabiner and B. H. Juang. An introduction to Hidden Markov Models. *IEEE Signal Processing Magazine*, 61(3):4–16, June 1986.