A Low-cost Spatiotemporal Data Collection System for Tennis

Silvia Vinyes Mora, George Barnett, Casper O. da Costa-Luis, Jose Garcia Maegli, Martin Ingram, James Neale and William J. Knottenbelt

Imperial College London, London, U.K.
silvia.vinyes-mora12@imperial.ac.uk, george.barnett10@imperial.ac.uk, casper.dcl@physics.org, jose.garcia-maegli14@imperial.ac.uk, martin.ingram14@imperial.ac.uk, james.neale14@imperial.ac.uk, wjk@imperial.ac.uk

Abstract

Technology has revolutionised tennis, especially the Hawk-Eye system which is now an integral part of professional tennis tournaments with the line-call challenge system. While such technologies are highly accurate, they are also equipment-intensive, costly and require substantial expertise to install on a court. Our aim is to investigate the feasibility of constructing a low-cost alternative from commodity hardware components and to investigate what compromise can be achieved between cost, accuracy and speed. The system that we implement is composed of four high frame rate, high definition grey scale cameras and a GPU-enabled desktop computer - a highly portable system that can be installed on court in minutes. The current system provides accurate spatio-temporal information about the players, the ball and their position relative to the court lines, and is able to replicate results in a 3D virtual environment.

1 Introduction

The integration of technology in sports has represented a new era in how matches are broadcast; the data that is collected from them and therefore the analysis that can be achieved. The state-of-the-art technology involved in officiating and collecting data in sports is the Hawk-Eye system [5]. Hawk-Eye is used in many sports and has become an integral part in professional tennis matches by the introduction of new rules regulating the line-call challenges. This system is extremely accurate but also highly sophisticated, comprising between 8 to 10 high speed cameras (more than 1000fps) and an extremely powerful computer. The fact that this technology is equipment-intensive, costly and requires expertise to install on a court restricts its availability to high profile venues of major tournaments. Alternatives exist but are similarly costly and equipment-intensive. Our aim is to develop a low-cost alternative from commodity hardware components able to replicate the events that occur on the court in a 3D virtual environment in real-time. Therefore, the specific hardware components have been selected based on our careful examination about the compromise that can be achieved between cost, accuracy and speed. The system that we propose is composed of four high frame rate, high definition greyscale cameras and a GPU-enabled desktop computer.

2 System Architecture

2.1 Hardware

The principal constraints when choosing the hardware equipment for this project are cost minimisation and portability. The system configuration is shown in Figure 1 and is described

in detail in this section. A clear image of the ball from at least two cameras is required at all times so that its 3D position can be inferred. To guarantee this, it was decided that the minimum number of cameras required is four, one at each corner of the court. The camera model chosen is the Basler Ace acA 1300-60gm, these are high resolution (1280 x 1024 pixels) and high frame rate (60 fps) monochrome cameras, therefore minimising motion blur. We have complemented them with low distortion Kowa LMVZ4411 lenses, which have dimensions matching the area to be captured (the standard tennis court dimensions). The cameras are connected to a four-port Gigabit Ethernet card through Gigabit Ethernet cables that have a dual function: deliver the camera capture at high speed to the computer and provide Power over Ethernet to the cameras, reducing cabling requirements. The computer used for processing is an HP Z620 Desktop Workstation with a fast 400GB PCI-E SSD and a GTX 980 nVidia graphics card. The system is unobtrusive and highly portable.

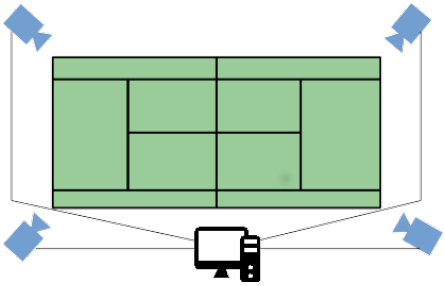


Figure 1: Configuration of the Detection System Hardware

2.2 Software

The software architecture is composed of three main layers as shown in Figure 2:

1. Frame capture: The Pylon Camera Software Suite was used to interface with the cameras and set properties such as exposure. An external 60 Hz signal generator was built to produce synchronized frame capture signals for the four cameras.
2. Frame processing: The computer vision part of the project is implemented in C++ using the OpenCV library [1]. It is organized as a multithreaded application in which player and ball are detected independently in each frame before the information is combined to obtain a 3D position.
3. Display: The data obtained from the frame processing module is fed into the data display module. This presents a 3D virtual environment in which the player and ball position are presented. It is a user-friendly display designed in UNITY, facilitating cross-platform portability.

3 Detection and tracking

3.1 Player Detection

The player detection process comprises two main tasks: background reconstruction and player detection itself. Figure 3 shows the intermediate images in detecting the player. For the

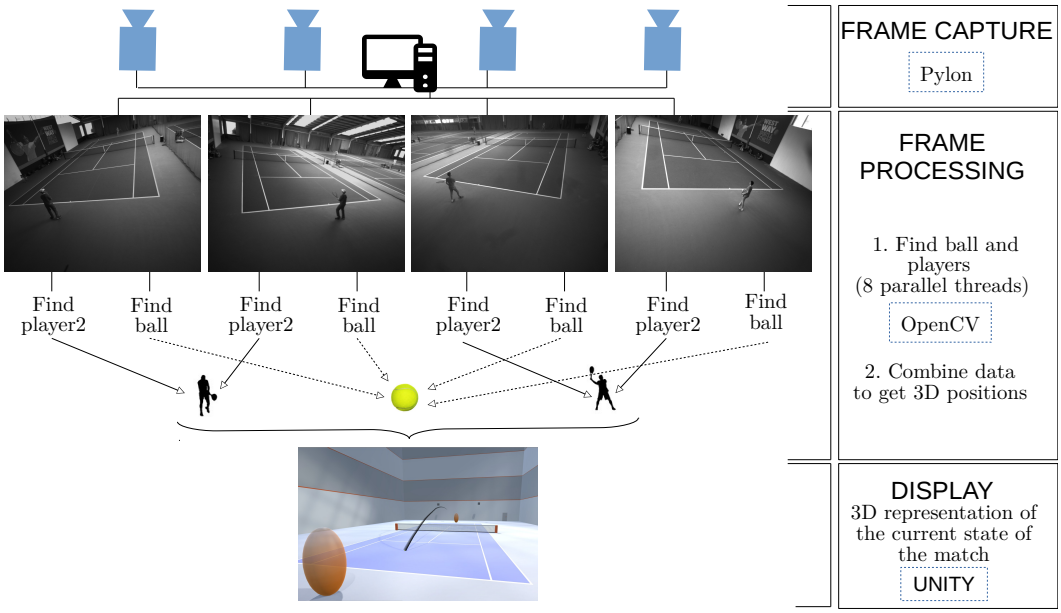
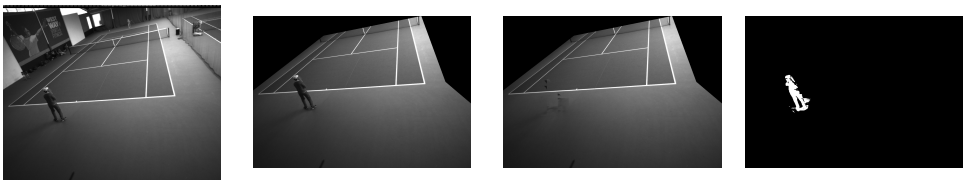


Figure 2: Software design

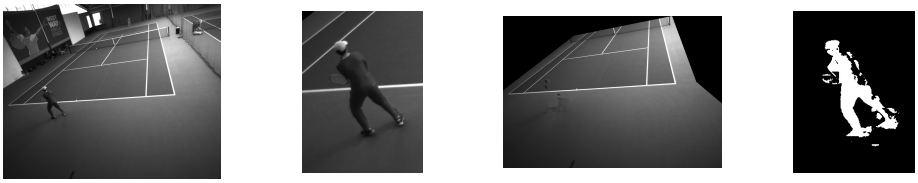
background reconstruction, the frame is divided in blocks. A block is incorporated to the background if the percentage of pixels that do not change over two frames is above a certain threshold. Blocks that do not satisfy this condition are discarded. After applying this process to a few frames, a complete background image is obtained for each of the cameras. Once this is complete, player detection can take place by selecting the foreground pixels within a region of interest (ROI), corresponding to the area in which the player can be located. For the frames in which the prior player position is unknown, the ROI for each player corresponds to their half court; detection is performed on these two areas separately. For the frames in which the prior positions are known, detection is performed only in areas within a given distance d of the player, d is defined according to the scale of the image and maximum speed of a player's movement. Once the collection of foreground pixels within the appropriate ROI is obtained, the center of mass is calculated using the moments of the shape and this is stored as the player position.

3.2 Ball Detection

The detection of the ball in a given frame n starts with the isolation of the pixels representing motion. These are stored in a matrix that will be referenced here as a motion matrix. The motion matrix is filled by computing the absolute pixel intensity difference between frames $n-1$ and frame n (backward difference). This is further refined by identifying bright points from the absolute difference of frames $n+1$ and $n-1$ (central difference) and setting these pixels to zero in the backward difference, thus eliminating the ball ghost from frame $n-1$. Next, contours of ball candidates are detected on the motion matrix using the OpenCV implementation [4]. To select the best candidate, a score is assigned to each potential ball according to its semblance



(a) Frame (b) Region of Interest (c) Background (d) Foreground



(e) Frame (f) Region of Interest (g) Background (h) Foreground

Figure 3: Player detection at the first iteration (a to d) and at a later stage (e to h)

to an ellipse [3]. Sequences of ball candidates which form possible trajectories are identified based on the method described in [2]. From these trajectory candidates, good trajectories are then selected using the following criteria: a quadratic fit of the points must have a negative first coefficient, the trajectory must comprise a large number of ball candidates (at least 8), and its ball candidates must fit an ellipse well on average. These good trajectories are then used to compute a best guess of the ball position in each frame. Ball detection is performed in parallel for the four cameras and the results are integrated by the next module to obtain the 3D position of the tennis ball.

3.3 3D Ball Position

Finding the 3D coordinates of a point, given its known position in images taken from two different views is a process known as triangulation. As shown in Figure 4, a given point y in a 2D image corresponds to a line R_y in 3D space. The 3D coordinates of that point are given by the intersection of the lines (R_y and $R_{y\prime}$ in Figure 4) corresponding to the two 2D points (y and $y\prime$) in the images taken from different cameras. However, noise in 2D coordinates (if we detect x and $x\prime$ instead of y and $y\prime$) generally means these lines (R_x and $R_{x\prime}$) do not intersect. Finding the 3D coordinate thus becomes a challenging problem for which different techniques can be applied. We have employed the mid-point method, in which the 3D point a is the value that minimizes the equation:

$$d(R_x, a)^2 + d(R_{x'}, a)^2 \text{ with } d(R, a) \text{ the euclidean distance between } R \text{ and } a$$

4 Results

As described, the ball and player are detected in each camera independently before the 3D coordinates are obtained via triangulation. Visual inspection of the 2D detection of the player and ball showed good accuracy across a large number of videos. Figure 5 shows the detailed analysis of the 2D results comparing our data to the ground truth for circa 5000 frames.

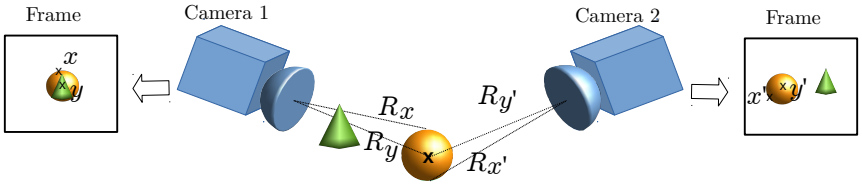
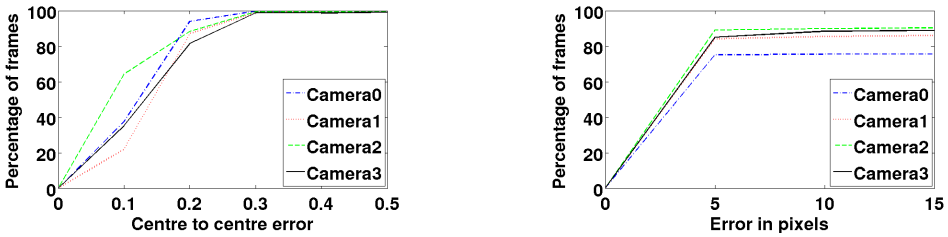


Figure 4: Triangulation



(a) Player detection results

(b) Ball detection results

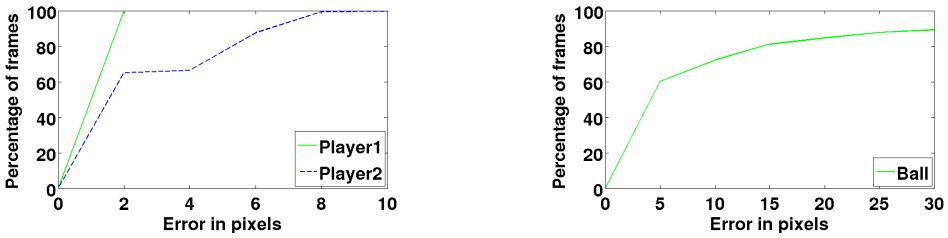
Figure 5: Analysis of the results for the player and ball detection in 2D

4.1 Player Detection

The player detection results in a 2D coordinate that represents the estimated center of mass of the player. The ground truth player data is stored as a rectangle surrounding the player. Note that each camera detects only the closest player, therefore there is a two to one camera to player ratio. The detected center of mass was within the ground truth bounding box for 100% of the analyzed frames. Figure 5a shows a fine grained analysis for each camera, depicting a curve representing the cumulative percentage of frames with a center-to-center error smaller than the x coordinate. This error represents the euclidean distance between the center of the ground truth bounding box and the detected center of mass difference. This distance is then represented as the percentage of the bounding box dimension to reflect that a given error distance of x pixels represents a larger error if the bounding box of the player is smaller. In general, errors are of the order of 15 pixels. This is a good result since the center of the bounding box does not correspond exactly to the center of mass of the player for many positions.

4.2 Ball Detection

The ball detection also produces a 2D coordinate, however the ball itself has a diameter of up to 10 pixels depending on the proximity to the camera. The objective is to determine the ball's center. Figure 5b shows the error in ball detection as the euclidean distance between the detected point and the ground truth for every frame in which a ball candidate was obtained. The ground truth data was obtained by manually annotating the same sequence of videos. Every curve in Figure 5b represents frames captured by a different camera, and it can be observed the error is less than 5 pixels for more than 75% of the frames.



(a) 3D Player detection results

(b) 3D Ball detection results

Figure 6: Analysis of the results for the player and ball detection in 3D

4.3 3D data

The results for the 2D can be accurately analyzed because ground truth data exists. However this is not the case for 3D data. As a first step, the 3D results have been visually inspected ensuring that the representation of the data in the 3D environment and the videos are consistent. Then we have also compared the ball 3D location obtained from the 2D ground-truth data to our 3D ball location estimates. It is important to note that the 3D data is produced for every single frame, while 2D data isn't. In fact, 3D data integrates data from all cameras and estimates the values for frames in which the ball was not detected. The same analysis was run for the player 3D location. Figure 6b shows the accuracy of our 3D results.

4.4 Running time analysis

The running time of the algorithm is 59.404 ± 0.595 fps, this is almost three times faster than when the code is run without using parallel threads (20.328 ± 0.083 fps) and does not include the I/O time since we run the experiments after buffering the frames.

5 Conclusion

We have designed a system able to collect 3D spatio-temporal data from a tennis match in real-time from commodity hardware. It is a low-cost alternative that has proven accurate results. In addition to this, its design is modular, with the capability of incorporating improvements both in terms of speed and accuracy. From the previously described results it would be interesting to investigate other triangulation methods and further increase the accuracy of the system.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Ciarán Ó Conaire, Philip Kelly, Damien Connaghan, and Noel E O'Connor. Tennissense: A platform for extracting semantic information from multi-camera tennis data. In *Digital Signal Processing, 2009 16th International Conference on*, pages 1–6. IEEE, 2009.

[3] Andrew W Fitzgibbon, Robert B Fisher, et al. A buyer's guide to conic fitting. *DAI Research paper*, 1996.

[4] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[5] Baodong Yan. Hawkeye technology using tennis match. *Computer Modelling and New Technologies*, 18(12):400–402, 2014.