Player and Ball Positional Tracking for Tennis and Cricket

Vivek Jayaram
Department of Computer Science
University of Washington
vjayaram@cs.washington.edu

Maruchi Kim
Department of Computer Science
University of Washington
mkimhj@cs.washington.edu

Aaliyah Hänni
Department of Data Science
University of Washington
fialaa@uw.edu

Abstract

In this project, we use state-of-the art computer vision algorithms to develop a comprehensive sports analytics system which operates on a single stationary video. Our system tackles two main problems: ball tracking and player tracking. Each of these tasks relies on several algorithms, including object detection, homography estimation, and tracking by detection. For each component of the algorithm, we investigate several different approaches and compare the performance. To showcase the generality of our system, we present qualitative and quantitative results on both cricket and tennis datasets.

1 Intro

Over the last two decades, player and ball tracking has rapidly grown in popularity at the highest levels of professional sport. Some examples of this include the use of Hawkeye Innovations technology or VirtualEye technology for reviews at international-level cricket tournaments and at tennis grand slams. Today, ball tracking is widely used across sports for decision reviews, viewer engagement, and automatic data collection. Amateur athletes, and even professionals who are not playing in the biggest tournaments, do not generally expect to play in an environment where such technology is enabled. However, the importance of this data has become clear during the age of growing data analysis and sports statistics.

The reason for the inaccessibility of such technologies at larger scale relates to their exorbitantly high prices, as well as complicated maintenance, installation and operational requirements. Often, these technologies require multiple (e.g., 6-10) high speed cameras operating at high frames per second (e.g., 340 fps), optic fiber cables connecting these cameras to a central hub and a number of operators ensuring accuracy and smooth operation. The broad market has much to gain from similar technology, even at lower accuracy, by using the derived data for performance improvement, scouting, and social media sharing.

In this paper, we focus on ball and player tracking from a stationary single camera feed, which could be captured by a mobile phone or a TV camera. Although existing systems can perform 3D ball tracking, we focus on detecting and tracking the ball in 2D. We can then produce real-world metrics, such as bounce location, by combining this 2D track with camera homography estimation. For player tracking, we also use frame level detections along with homography estimation to get the players' positions on the court at any given moment. The output of our system is the 3D player position, 3D bounce location of the ball, and 2D ball position in every frame.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Figure 1: (Left) An example frame from the cricket dataset with the ball location highlighted in green. (Right) An example frame from the tennis dataset with the ball location highlighted in red.

2 Related Works

2.1 Commercial Multi-Camera Solutions

The most well known ball and player tracking are those which are used at the highest level of sports. These include Hawkeye and VirtualEye which are the leaders in tennis and cricket, as well as Second Spectrum, SportsRadar, and Tracab for other sports such as basketball, soccer, and cricket. These systems use multiple cameras to triangulate the ball and player positions at a specific moment ((Owens et al., 2003)). In general, the ball is detected in each frame of each camera independently, then combined using camera calibrations and a triangulation algorithm such as the one described in ((Wu et al., 2020)).

2.2 Commercial Single-Camera Solutions

Recently there have been a number of single-camera products attempting to use available hardware, such as mobile phones or broadcast footage, to perform similar tracking and analytics. Existing products for tennis include SwingVision and ArtLabs. In cricket there have been several companies including MachineRoad, PitchVision, and Fulltrack which attempt to do ball tracking from a single camera.

2.3 Published Works

Among publicly available works, TrackNet ((Huang et al., 2019)), is the most similar to ours in the domain of tennis. They develop a multi-frame CNN to identify the tennis ball in each frame, but their backbone is based on VGG whereas ours is based on a UNet architecture. Other tennis works on ball tracking include Yoo et al. ((2018)), Ekinci and M. ((2008)), and Archana and Geetha ((2015)).

For cricket, Udit Arora ((2017)) uses an SVM trained on HOG features to identify the cricket ball in each frame, then use a temporal smoothing algorithm to reject false positives. Other cricket works include Kumar ((2010)), and Liu et al. ((2011)).

3 Dataset

3.1 Cricket Dataset

Our dataset for cricket comprises of 1040 videos of 3 seconds each. These videos are at 1280 x 720 portrait resolution and 60 frames-per-second. Each video contains one delivery, which is similar to a baseball pitch, which was captured by the Fulltrack AI mobile capture system. In addition to the videos, the ball position in certain frames is provided. This is formatted as a 2D pixel location and radius, since the ball will be a circle unless motion blur is large. Notably, the ball location is not provided for every single frame. In total, we have 15,640 frames for training and 2,330 frames for testing.

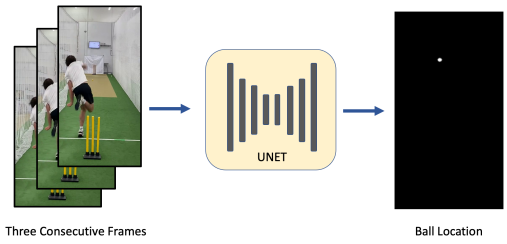

Figure 2: Our multi-frame UNet model. Three consecutive RGB frames are concatenated and fed through the network. This produces a heatmap of the ball location for the central frame which can then be detected with a threshold and non-max suppression

3.2 Tennis Dataset

The tennis dataset contains 10 complete tennis matches, retrieved from broadcast videos. The video resolution is 1280 x 720 and 30 fps. Each game is split into 10 clips and each clip contains approximately 2000 frames. Every frame corresponds to a row in the labeled data set, containing the frame ball location in x, y pixel coordinates, as well as the associated action (bounce, hit, fly), and visibility. Each frame was manually labeled by human volunteers, using a custom developed labeling tool.

4 Ball Tracking

Our ball detection algorithm attempts to create a trajectory of the ball across frames in the video. To do that, we first train an object detector to localize the ball in each frame. We then employ a graph-based tracking algorithm to create a track across frames which can handle false positives and missed detections.

4.1 Object Detection

We evaluate several different object detectors for the purpose of detecting the ball in a single frame. The first is a multi-frame UNet where three consecutive RGB frames are concatenated and fed to the network. The network outputs a single channel tensor with half the spatial dimensions, with all values between 0 and 1. For example, in cricket the input tensor has dimensions $9 \times 1280 \times 720$ with an output size of 640×360. For training, we create a heatmap with all zeros except in the location of the ball where the value is 1. The network is then trained with a simple pixel-wise binary cross entropy loss. We demonstrate this network in figure 2As a simple comparison, we also train a single-channel UNet which takes only an RGB image.

To produce a detection, we find pixels above some threshold value, then perform non-max suppression to avoid nearby detections. In table 1, we show the precision and recall of this method on the test set.

4.2 Object Tracking

Although the ball detector does a good job picking up the ball on each frame, it is prone to false detections as well as missing detections. We use a graph based algorithm to produce a temporally consistent track. We create a graph as follows: Each detection is a node in the graph. Suppose we have two ball detections x and y. They are connected in the graph if they are close in space and also

Table 1: Object Detector Results - Recall (Precision)

	Single Frame U-Net	Multi Frame U-Net
Cricket	77.6 (55.2)	99.5 (92.1)
Tennis	68.0 (25.5)	92.7 (42.4)

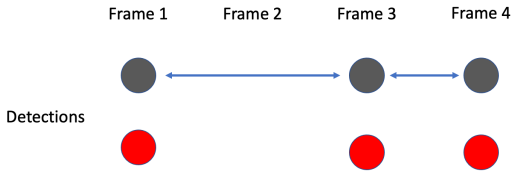

Figure 3: We construct a graph where detections are connected if 1) they are spatially close to each other 2) they are temporally close. False detections here are shown in red and are not connected because they are usually far away from other detections. Frame 2 shows a frame with missing detections. We find the longest path in this graph using dynamic programming.

Figure 4: A comparison of the ball detections (left) with the ball tracking output (right). The tracking result produces a temporally consistent track which handles false and missing detections.

in nearby frames. Formally, that means $||x - y|| < \epsilon_1$ and $|x_t - y_t| < \epsilon_2$. A visualization of such a graph is shown in figure 3. After constructing such a graph, we can find the longest path using dynamic programming. This longest path corresponds to the longest trajectory of likely ball locations. Assuming that there are more true detections than false detections, this will give us the trajectory of the ball. We consider all detections equally, but it could be possible to weight the detections by their detection score when constructing the graph. A visualization of the results after finding the longest path is shown in figure 4

Formally, the algorithm works as follows. We look detection in reverse order of frame index, starting at the last observed detection t_{max}. This node has score 1. Now we continue looking at nodes in previous frames. For a node x at time x_t, consider the connected nodes we already visited, $y \in x$.edges such that $y_t > x_t$. The score of x is $1 + max(y.\text{score})$. By iterating through all the nodes in reverse chronological order, we build up the longest path of the ball which is the longest connected path in this graph.

5 Player Tracking

Our player tracking algorithm attempts to project the x,y coordinates of both tennis players onto an idealized bird's eye view of a tennis court. To do this, we first detect all objects across a series of frames of video using YOLO. We then extract the locations of the players, while taking care to filter out false positives (i.e. other 'person' objects) and interpolate across frames where a player failed to be detected.

To determine the location of the players, we use a lightweight version of YOLO (TinyYOLO) which came pre-trained. We input a list of images (representing the original video) into the detector, and output all detections. Below is an example output from a single frame of video:

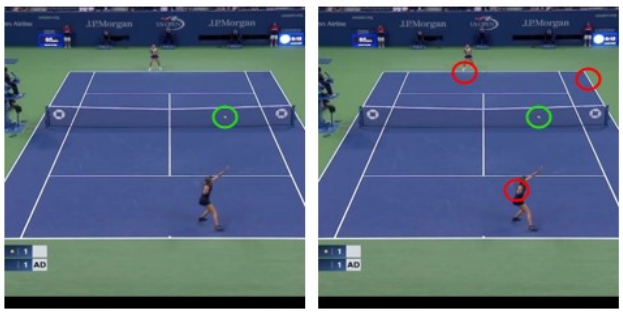

Figure 5: An example frame taken from our test set of tennis matches (left), with the tennis ball highlighted in green to illustrate the location. The output of the Multi Frame U-Net (right), with true positives highlighted in green, and false positives in red. Observe that we successfully identified the location of the tennis ball, but that the model is prone to detecting multiple false positives, motivating the implementation of the graphing algorithm.

```
person: 42% (left_x:  299   top_y:  503   width:   49   height:   63)
person: 26% (left_x:  669   top_y:  132   width:   23   height:   27)
person: 28% (left_x:  672   top_y:  112   width:   18   height:   38)
person: 28% (left_x: 1169   top_y:  305   width:   20   height:   33)
```

5.1 Filtering and Interpolation

While TinyYOLO performs well in outputting the coordinates of every person object, our player tracking algorithm necessitates a method to filter out non-player objects, and also interpolate between frames where a player object failed to be detected.

To detect non-player objects, we assume that each respective tennis player can not travel more than a predefined threshold of distance between frames (i.e. each player can not 'teleport' to an inconceivable location within a frame). This helps our player tracking system to filter out umpires, line judges, and ball boys and ball girls.

One final step of pre-processing is to filter between frames where a player object was failed to be detected. In the event that a player was failed to be detected, we interpolate between frames to estimate the player position. Below is high-level pseudocode of the interpolation step.

```
for every frame in video:
  if (player not detected within threshold):
    last_known_position = previous_position
    threshold += x
    interpolate = true
  else if (interpolate):
    interpolate between last_known_position and frame
  else:
    interpolate = false
```

5.2 Homography Estimation and Video Output

To estimate the homography between the video and a 2D court. We utilize OpenCV. We choose 4 desired points in the video frame (4 corners of the tennis court), and 4 points in the 2D court image to calculate the homography matrix.

Finally, with a list of player positions and ball positions, we project x,y coordinates of both players and the ball using our calculated homography matrix onto a 2D tennis court image. We stitch together each frame, and output a video which can be seen in Figure 6. Original video and output video can be viewed with the following links:

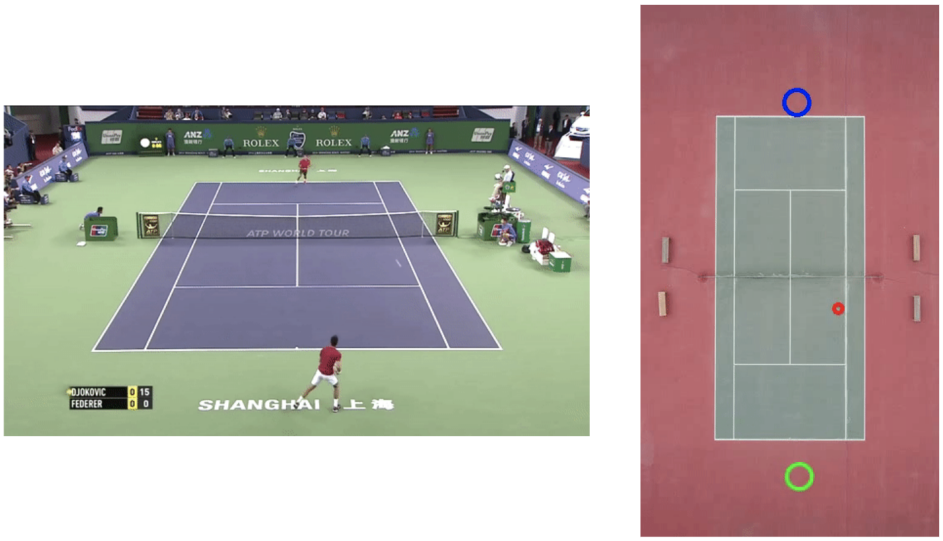

Figure 6: Original video frame (left) and output video frame of our player and ball tracking system (right). Blue and Green circles represent the players, and the red circle represents the ball.

Original video: https://drive.google.com/file/d/1bs-v1NfXcVKP7ix_wY_NRT9iK700MZvf/view?usp=sharing

Output video: https://drive.google.com/file/d/1XPGS7w8zLBHX5ReKs5sDqJ3xUlgWUsOG/view?usp=sharing

6 Future Works

6.1 Action Detections

We would like to expand our work by not only computing the ball location, but also key actions. This can be accomplished by utilizing the labels of the datasets to predict actions associated with the ball. For example, training a model to capture the ball state, such as 'bounce', 'hit', or 'fly', as well as predicting overall game states, such as when a score is made.

6.2 Sports Recommendations

Building on our findings, we would have liked to provide an application of our project in the form of sports recommendations. An example of how we would implement this is by selecting clips where a player hits the ball, and compare their positioning to that of professional player positioning in successful games. This recommendation system would then be able to provide instant feedback to sport playing users on how they can change their stance or position to obtain an enhancement in performance.

6.3 Other Sports

To test the scalability of our models and approaches, we would like to apply this same methodology to other sports games, such as soccer, baseball, and hockey. By obtaining other labeled data set or by using our trained models, and measure how well they can perform in zero and few shot learning.

References

M. Archana and M. K. Geetha. Object detection and tracking based on trajectory in broadcast tennis video. *Procedia Computer Science*, 58:225–232, 2015. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2015.08.060. URL https://www.sciencedirect.com/science/

article/pii/S1877050915021717. Second International Symposium on Computer Vision and the Internet (VisionNet'15).

B. Ekinci and G. M. A ball tracking system for offline tennis videos. pages 45–48, 2008. doi: 10.5555/1980844.1980854.

Y.-C. Huang, I.-N. Liao, C.-H. Chen, T.-U. Ik, and W.-C. Peng. Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications. *CoRR*, abs/1907.03698, 2019. URL http://arxiv.org/abs/1907.03698.

P. A. Kumar. An efficient ball detection framework for cricket. 2010.

D. Liu, M.-L. Shyu, Q. Zhu, and S.-C. Chen. Moving object detection under object occlusion situations in video sequences. pages 271–278, 12 2011. doi: 10.1109/ISM.2011.50.

N. Owens, C. Harris, and C. Stennett. Hawk-eye tennis system. pages 182–185, 2003. doi: 10.1049/cp:20030517.

S. S. T. S. Udit Arora, Sohit Verma1. Cricket umpire assistance and ball tracking system1 using a single smartphone camera. *PeerJ Preprints*, 2017. doi: 10.7287/peerj.preprints.3402v1. URL https://www.readcube.com/articles/10.7287%2Fpeerj.preprints.3402v1.

W. Wu, M. Xu, Q. Liang, L. Mei, and Y. Peng. Multi-camera 3d ball tracking framework for sports video. *IET Image Processing*, 14(15):3751–3761, 2020. doi: https://doi.org/10.1049/iet-ipr.2020.0757. URL https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-ipr.2020.0757.

W.-S. Yoo, Z. Jones, H. Atsbaha, and D. Wingfield. Painless tennis ball tracking system. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 783–784, 2018. doi: 10.1109/COMPSAC.2018.00118.