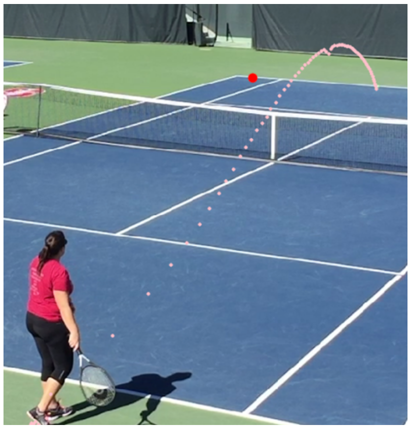
Tennis Ball Tracking:
3D Trajectory Estimation using Smartphone Videos

Megan Fazio[1], Kyle Fisher[1], and Tori Fujinami[1]

***Abstract*— In this project, we present a system which processes recorded stereo smartphone videos to estimate the 3D trajectory of a tennis ball as it moves during play. Given recorded video footage collected by two smartphones mounted on tripods, our system combines various image processing techniques to interpret the video frames and reconstruct the ball trajectory. To do this, our system locates point correspondences of the court in image space, determines the camera locations relative to a globally-defined origin, and estimates the ball location using multiple-view geometry and state estimation filtering. To this end, we employ image processing concepts including image segmentation, morphological image processing, the Hough transform, moments analysis, blob analysis, camera calibration, and multiple-view geometry.**

I. Introduction

Tennis is a sport which is enjoyed by millions of people around the world. For players practicing to improve their skills, useful information is gathered from studying the 3D trajectory of the ball as it moves during play. Measuring this trajectory typically requires highly-specialized recording equipment and careful calibration. At the professional level, the Hawk-Eye system is employed for line calls and replay analysis. While this system is too expensive to setup for most recreational use, a less expensive system with similar functionality and potentially lower accuracy is desirable. In our project, we make trajectory estimation more consumer-accessible. We present a ball trajectory-measuring system which relies only on two smartphone cameras and our own algorithm for automatically interpreting the recorded stereo footage. Given two recorded views of the tennis game, our system outputs frame-by-frame data including the balls speed and estimated 3D position with an average error of 0.5m.



II. Related Work

Several papers have been written about algorithms and techniques to track a tennis ball in videos, both in post-processing and in real-time. Qazi et al. developed an algorithm for detecting and tracking a tennis ball in videos of tennis matches filmed from a quadcopter [1]. Lyu et al. developed algorithms for stabilizing videos and use a method called random forest segmentation for extracting possible ball locations in the image [2]. Archana et al. propose a new algorithm for ball detection that utilizes frame difference, logical AND operations, thresholding, and dilation to track the motion of the tennis ball in broadcast tennis videos [3]. Yan et al. use motion segmentation and foreground blob detection to determine probable locations for the tennis ball, modeling the ball with a LTI-system and using a particle filter to determine the most probable location for the ball given the images [4]. Ekinci et al. process videos from a fixed camera to extract the background of an image and generate ball location candidates, then use a Kalman Filter to narrow down the candidates to the most probable ball location [5]. Yu et al. use limited visual cues to track the tennis ball more accurately in broadcast tennis videos and described an algorithm for inferring ball position based on previous and later frames [6].

III. Implementation

A. Overview

Our system is implemented as a computer program that accepts synchronized stereo video footage as input and outputs the approximated speed and 3D position of the tennis ball at each frame of the videos. For previewing the outputted trajectory, we provide a visualization tool which renders split-screen videos showing the reconstructed ball path annotated on the stereo input frames, as well as two computer-generated plots: a top-down orthographic view, and a perspectivized 3D plot.

B. High-Level System Design

Once provided with synchronized stereo footage of a tennis game, our system acts in a pipeline-like manner, passing intermediate results from one stage to the next. The first stage of the pipeline is frame segmentation, where the corners of the court and a set of candidate ball positions are found. Next, the program determines the position and orientation of the camera in a 3D coordinate system whose origin lies at the center of the court. This is done using the court's four corners as point correspondences between image and world space. With knowledge of the camera

[1] Department of Electrical Engineering, Stanford University, MS students {mefazio, kefister, fujinam2} @stanford.edu

position and orientation, the system then calculates nearest ray-intersections using all possible combinations of candidate ball positions from the two cameras. Finally, the convergent near-intersection points are passed through a Kalman Filter to estimate the true ball trajectory from the set of candidate ball positions, and the estimated trajectory is written to a CSV file.

C. Data Collection

In our experiments, we collected video footage manually using two smartphones with the same camera: an Apple iPhone 6 and an Apple iPhone 6 Plus. Each camera is set to film at 60fps and 1920 x 1080 pixels resolution. We filmed 20 groundstrokes in stereo at the Taube Tennis Center at Stanford in clear-skied, daylight conditions. We positioned the two smartphones at elevated locations in the stadium's seating area and aligned each camera to a respective sideline of the court. Each camera was aimed toward the center of the court as shown in Figure 1, with the entire court in view of the camera to ensure that the court segmentation succeeded. To simplify synchronization, we filmed a single video continuously from each camera during the session.

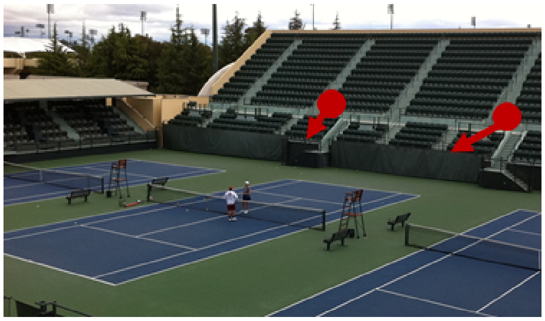


Fig. 1. iPhone camera setup

After filming, we manually cut the video into clips with one hit ball per clip. For each test hit, the player held the ball out with the non-racquet hand at approximately waist height and released the ball so that it bounced on the court and rose back up before being hit with the racquet. We trimmed the clips to start with the first frame where the tennis ball bounced before being hit, and end when the ball either left the frame or impacted the rear wall. The first frame of a clip from each of the cameras is shown in Figure 2.

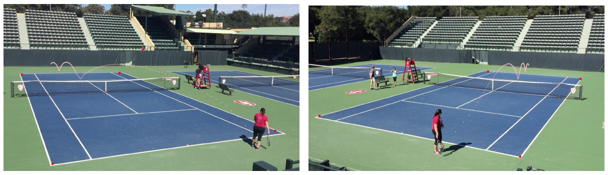


Fig. 2. Temporal synchronization of video clips

D. Camera Calibration

For accurate 3D position estimation, our system requires knowledge of the intrinsic camera parameters, including focal length, optical center, radial distortion coefficients, and tangential distortion coefficients. To estimate these parameters for our smartphones, we record video of a checkerboard pattern calibration target shown in Figure 3 from a variety of different angles and across various locations in the field of view of the camera. We extract several representative frames from each calibration video and use these as inputs to the MATLAB Camera Calibrator application, which outputs each of the parameters required by our system.

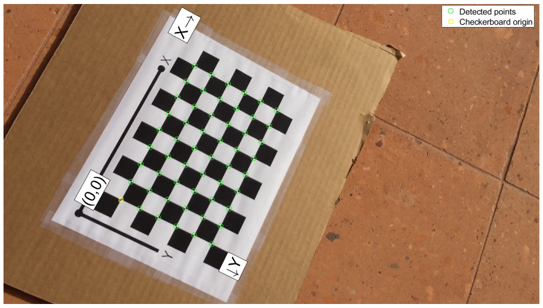


Fig. 3. Camera calibration target

E. Court Segmentation

To solve for the camera's position and orientation relative to the world coordinate frame, our program must first estimate the position of the court in image space. In our implementation, we focus on searching the input frames for the court's four corners – where the four outermost white lines meet. These four pixel locations form a unique and complete description of the court's position in the frame.

We assume that the center of the court (where the net meets the center service line) lies approximately at the center of the frame. Under this assumption, we can estimate the dominant color of the court using a crop section. This is done by extracting a rectangular window from the center of the frame and computing an HSV histogram to find the peak color intensity as seen in Figure 4. In our footage, the dominant court color is a dark shade of blue.

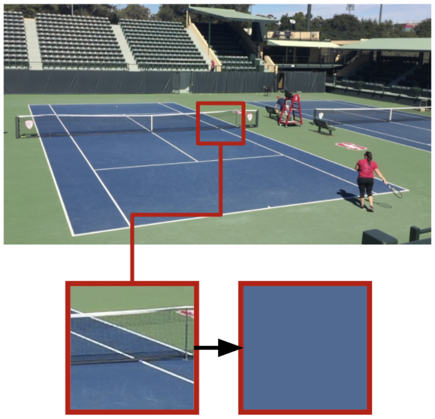


Fig. 4. A rectangular window is cropped from the center of the image and used to find the dominant color which is then considered the court color

With the dominant color extracted, we perform HSV thresholding on the image to obtain a mask containing only regions where the color nearly matches the dominant court color. This is displayed in Figure 5. Since there may be other court-colored objects in the scene, such as additional courts in the background, we filter out smaller regions using a series of morphological operations and small-region removal. The result is a mask which consists of the approximate closed area occupied by the court in the frame as depicted in Figure 6.

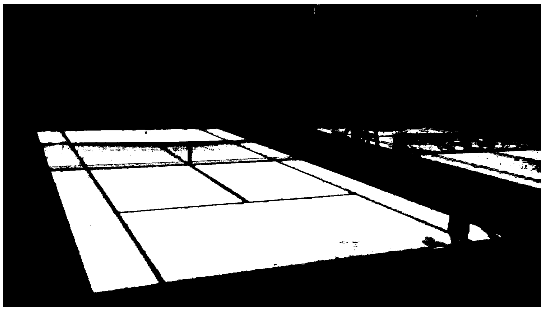


Fig. 5. Unfiltered HSV mask of the court

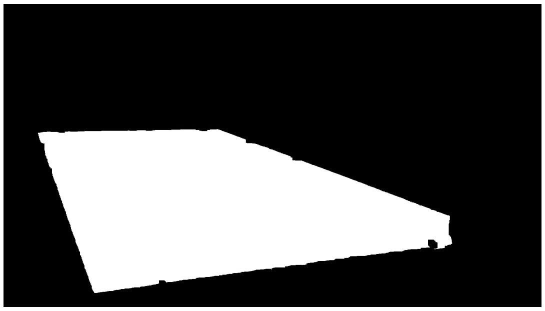


Fig. 6. Filtered HSV mask of the court

Next, canny edge detection is applied to this mask to form an outline of the court (Figure 7). Using the Hough transform, we identify the prominent lines in the outline (Figure 8), and record all intersections that occur between the Hough lines in a separate mask. This mask is dilated substantially, so that clusters of intersections become connected components.

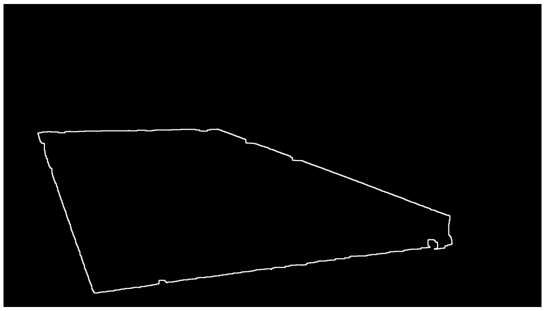


Fig. 7. Canny Edge Detection

Then, the four largest connected components in the mask are taken to lie at the center of the court's four corners. Their centroids are calculated, and the resulting four X-Y pixel coordinates are sorted by their approximate relative locations. The final result is displayed in Figure 9.

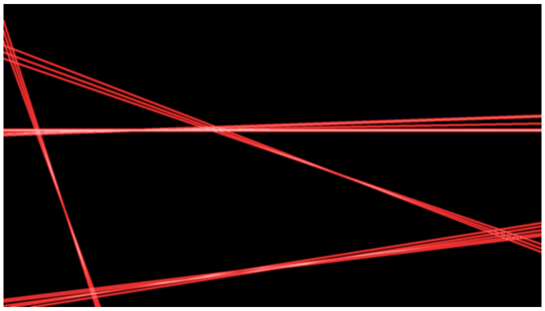


Fig. 8. Hough transform lines

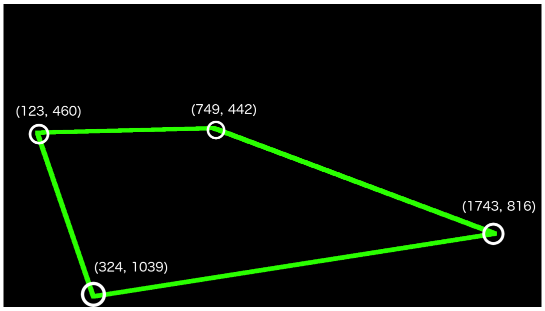


Fig. 9. Final court corners detected

F. Ball Segmentation

Ball segmentation is done by combining several different image masks. All image masks use frame differences between the current frame and the frame 5 frames ahead. Five frames ahead was chosen so that the future ball has very small probability of overlapping with the ball in the current frame, thus enabling frame differences to be used. The first mask takes both frames in the RGB space and converts them into HSV space. Each frame in HSV space is thresholded to create a binary mask based on ball color, where the threshold is approximately the mean HSV value for the color of a tennis ball. To reduce the false positives generated by the background of each image, a difference is taken between successive frame masks. An XOR can be performed to get rid of any common background pixels, and an AND with the original HSV mask will return only the ball in the first frame while maintaining the cleared background.

The second mask takes both frames in the RGB space and computes the difference of the frames in the RGB space, which is thresholded to create a new binary mask. The eccentricity of the large difference regions is then analyzed to determine approximately ball-shaped blobs of reasonable size. The remaining blobs are additionally filtered by an AND with a weak HSV mask so that only the blob corresponding to the ball in the original frame remains.

The third mask considers cornerness. Frames in RGB space are input and the difference is calculated. The difference is then converted to grayscale and the weak HSV mask is ANDed to remove large chunks of the background and the second ball. Harris cornerness is then measured on the remaining differences and the binarized output is used as a third ball mask.

Combining the outputs of the three masks, we consider pixels that appear in at least two of the three masks to be a ball candidate pixel. We refer to the final, combined mask as the frame's ”ball segmentation mask”. An example of a ball segmentation mask is depicted in Figure 10. The lone point just to the right of the center corresponds to the true ball, in this particular case.

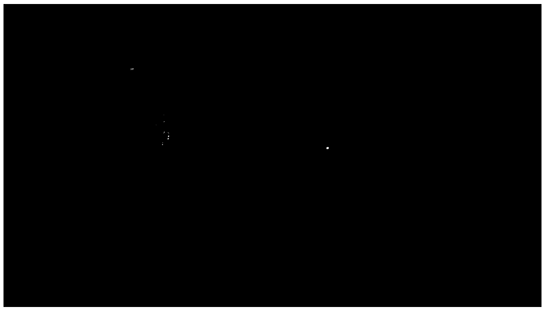


Fig. 10. Ball segmentation mask

G. Ray Intersection

Using the camera calibration parameters and pixel coordinates of the court corners, a linear pinhole camera model and court plane homography mapping between pixel coordinates and world locations is created for each camera. We define our world coordinate system with a Y-up convention, setting the origin at the exact center of the court. We let X and Z lie the plane of the court. The rays passing through the centroid of each ball candidate in the ball mask are produced and then intersected (see Figure 11). By comparing every combination of camera-to-ball rays from each of the stereo frames, we check the validity of a ball candidate by calculating the world location of ray-pair intersections and assign a confidence to a pair of convergent rays by measuring their nearest-intersection distance. These metrics are more closely considered in the state estimation filtering described in the next section.

H. State Estimation

The final step takes ray intersections and determines the most likely ball position in world coordinates and real-world ball velocity via a Kalman filter with unknown correspondence. The Kalman filter is intended to utilize system model knowledge and measurements with the understanding that both are imperfect by combining them in a two-step predict-update process. If model predictions and measurement updates match closely, then the filter is more certain that the estimate of the current state is more likely to be correct. Unknown correspondence means that in performing the Kalman filter update step, the exact measurement corresponding to the state that the filter is trying to estimate is unknown.

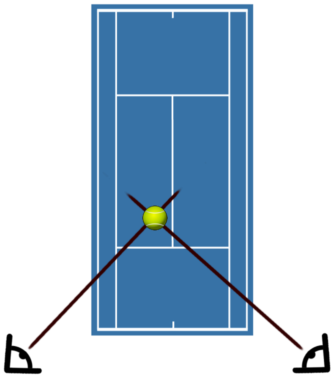


Fig. 11. Depiction of triangulation process for ball localization in world coordinates via ray intersection

The state for this Kalman filter is the ball XYZ position and XYZ velocities. The Kalman filter state is initialized to all zeros. Knowing this is incorrect, but not knowing the exact initial location of the ball, we can fix this and ensure the Kalman filter converges by setting the initial uncertainty (state covariance matrix), to be very large.

Since the state is ball position and velocity, we can devise a very simple linear physics system as the state prediction model. We know the mass of the ball and acceleration due to gravity (acting in the -Y direction). There is technically also a force and acceleration in the X and Z directions in the moment at which the player hits the ball. However, from our current image segmentation, there is no way to determine time at which the ball is hit and the input force or direction corresponding to this hit. So, for simplicity, we can treat this as a point-mass adhering to just gravity and assume constant X and Z velocity after the ball has been hit. The basic physics equations are listed in Equation 1. In these equations, g represents acceleration due to gravity and Δt is the time between frames so $1/framerate$. This can be rewritten as single linear equation with a state vector as seen in Equation 2. $x_t \in \mathbb{R}^6$ is the state vector where XYZ position and XYZ velocity is stacked respectively. w_t is model noise which is assumed to be normally, independently and identically distributed at each time step. Equation 2 forms the basis of the prediction step for the state in the Kalman filter.

$$\begin{aligned} x_{t+1} &= x_t + v_{x,t}\Delta t \\ y_{t+1} &= y_t + v_{y,t}\Delta t + \frac{1}{2}g(\Delta t)^2 \\ z_{t+1} &= z_t + v_{z,t}\Delta t \\ v_{x,t+1} &= v_{x,t} \\ v_{y,t+1} &= v_{y,t} + g\Delta t \\ v_{z,t+1} &= v_{z,t} \end{aligned} \tag{1}$$

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t + w_t \\
A &= \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
B &= \begin{bmatrix} 0 \\ \frac{1}{2}(\Delta t)^2 \\ 0 \\ 0 \\ \Delta t \\ 0 \end{bmatrix} \\
u_t &= g
\end{aligned} \tag{2}
$$

The measurement model assumes measurement of the partial state directly. Specifically, a measurement is assumed to directly measure ball XYZ position. The measurement model equation is expressed in Equation 3 where y_0, y_1, and y_2 correspond to measured XYZ position respectively, and v_t is measurement noise also assumed to be normally, independently, and identically distributed. Ray candidates are pruned initially based on ray intersection distance (the closest distance between the rays). We use this threshold value as an estimate for measurement noise. We additionally prune rays that are too far out of the court range in the X,Y and Z directions because those points are probably noise instead of a ball in play within some proximity of the court. However, with each set of ray intersections, the filter still receives multiple measurements. From this set, at most only one measurement corresponds to the measurement of the ball location.

$$
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}_t + v_t \tag{3}
$$

Often, correspondence is determined by the minimum Mahalonobis distance which scales the L2-norm of the difference between state estimate and measurement by the covariance matrix. However, for this project, measurement correspondence is is based on a series of cases. Initially the ball location is guessed, so any measurements near the ball estimate are probably not the ball. This is why we do not use the Mahalonobis distance. Instead, if the uncertainty measured as the Frobenius norm of the covariance matrix is very large, then the Kalman filter update is performed on the measurement with the smallest ray intersection distance. This is most likely to be a valid point that appears in both images, and ideally, only real balls produce valid ray intersections and move in such a way that follows the basic physics model.

As it turns out, often the moving player also produces a number of valid ray intersections, but these may not necessarily move in the pattern of the ball. This is an issue particularly if the frame does not pick up the real ball among the set of candidate measurements. To get around this, if the uncertainty is small enough, then the Kalman filter assumes the current estimate of the state is close to correct. So, the update considers the measurement closest to the predicted ball location for that time step. If this measurement is still too far away from the predicted ball location, then it is considered invalid and the update step is skipped under the assumption that no measurements of the ball were actually received. This case outputs the predicted ball location as the current state estimate. The final case for determining correspondence aims to fill the gap between having high and low certainty in the ball location. For any case where certainty is not small enough to more fully trust the current state estimate but not large enough to consider the state estimate completely wrong, the Kalman filter chooses the measurement that minimizes the uncertainty as the most likely ball measurement.

IV. Results

A. *Court Corner Location Accuracy*

To test the court corner-finding stage, we collected additional images of tennis courts from the web. These images were deemed appropriate tests for our algorithm, since they were captured in evenly-lit environments and with the entire court in frame. Figure 12 shows the complete test set used in our experiments, with the outputted corner locations superimposed as crosshair markers.

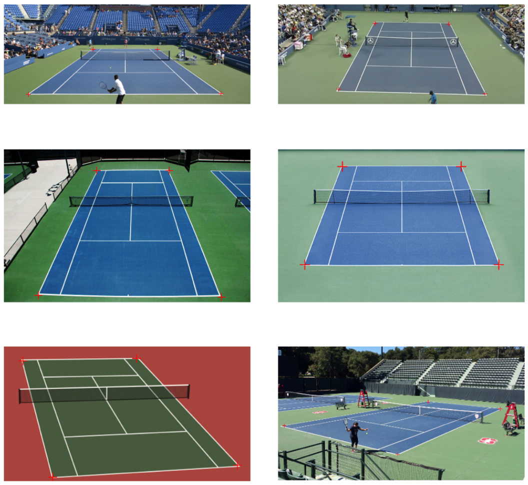


Fig. 12. Complete image set for testing corner location accuracy

In all cases, the court's corners are correctly identified, but always with some small positional error. For these tests, we outputted the frame with crosshairs marking the court corners so that the accuracy could be inspected. Several zoomed-in examples of this can be seen in Figure 13. Using manually-determined ground truth of true corner positions, we recorded the image-space distance error of the pixel positions outputted. For each input image, we report the average absolute error over the image's four detected corner locations in Table I.

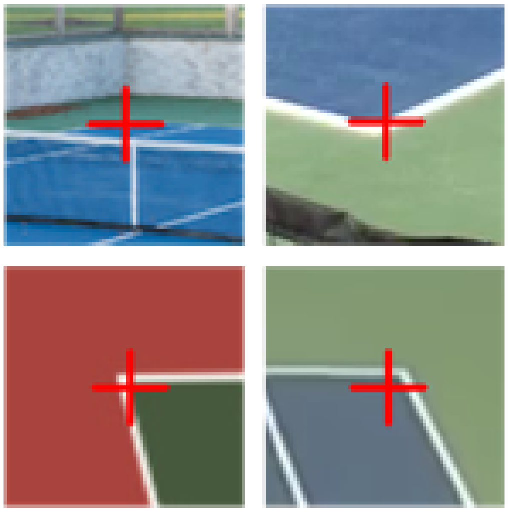


Fig. 13. Inspecting court corner accuracy with crosshairs

TABLE I

POSITIONAL ERROR OF COURT CORNER LOCATIONS IN PIXELS

Test Number	Input Resolution (px)	Average L2 Error (px)
1	916x558	6.6
2	1920x1080	13.7
3	768x570	3.5
4	1000x666	6.9
5	1630x808	12.3
6	1564x702	4.9

B. Ball Tracking Accuracy

Since the ball-segmentation step produces a variable number of candidate ball positions, the ray-intersection step also produces a number of candidate world positions for the ball. Ideally, we would see exactly one 2D candidate ball position in each mask, and these positions would form two rays in world space that would converge to an exact point. Since this is not the case, it is interesting to study the actual number of candidates outputted in each step.

In our experiments, we observe that ball segmentation on a single frame produces an average of 8.5 candidate ball positions per frame. A histogram showing the number of such positions outputted over the course of a test video is shown in Figure 14.

The vast majority of the 2D ball candidate pairs are filtered out because they form ray pairs that are considered too divergent. While ideally there would only be one convergent pair of rays, oftentimes we observe many. Figure 15 shows the number of 3D ball candidate positions we observe over the course of a test video. 40.2% of the time, we observe exactly one 3D ball candidate, while 6.9% of the time, we observe zero candidate positions. In all other frames, we observe multiple candidates. Fortunately, there is sufficient useful information for the Kalman filter to recognize the true location of the ball (and to predict its position when the ball is not found).

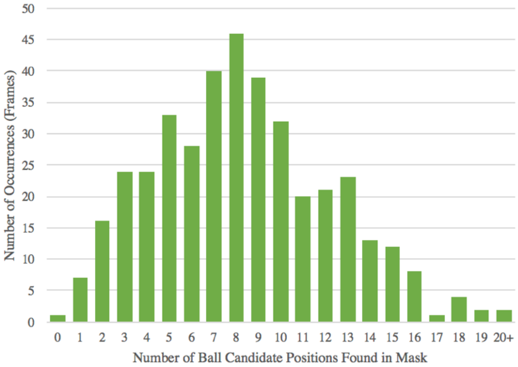


Fig. 14. Histogram showing the number of ball positions found during image segmentation. On average, a mask will contain 8.5 candidate ball positions per frame.

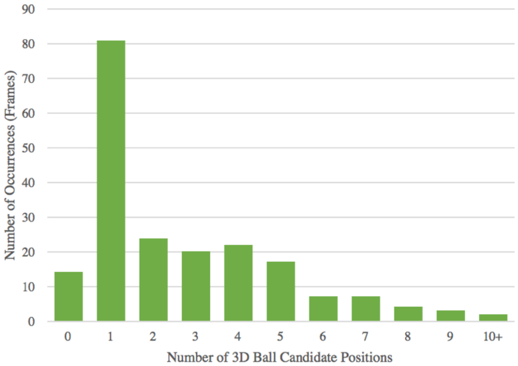


Fig. 15. Histogram of number of convergent ray intersections passed into the Kalman filter per frame

C. Ball Position Estimation Accuracy

To evaluate the accuracy of our 2D-to-3D spatial estimation, we conducted 10 tests where we fixed the ball at various known positions around the court and began filming. As ground truth for these positions, we used standard tennis court dimensions. By comparing these known positions with our own estimated 3D position, we recorded the error for each test position in Table II. Note that our system performs worst in the Z direction.

TABLE II

POSITIONAL ERROR IN BALL LOCATIONS IN METERS

Test Position	X Error	Y Error	Z Error	L2 Error
corner 1	-0.16	0.12	0.85	0.87
corner 2	0.07	0.12	0.50	0.52
corner 3	0.09	0.06	0.39	0.41
corner 4	-0.02	-0.02	0.10	0.10
left net post	-0.19	0.15	0.82	0.85
right net post	0.21	0.20	0.79	0.85
net center	-0.06	0.28	0.60	0.67
court center	-0.06	0.05	0.44	0.45
rear center mark	-0.08	0.12	0.47	0.49
front center mark	0.05	-0.02	0.15	0.16
Maximum Error	0.21	0.28	0.85	0.87

D. Qualitative Results

In order to visually evaluate our court and ball segmentation, we display each frame of the video and superimpose markers on the court corners and the ball. We display each frame in succession and leave the markers for the ball in previous frames so that the path of the ball is clear in the video. In Figure 16 below, the corners of the court are marked with red circles and the sequential positions of the ball are marked with pink circles. As can be seen in the image below, our court detection algorithm correctly identifies the corners of the court and the ball detection algorithm tracks the path of the ball with sufficient accuracy for visually analyzing the shot.

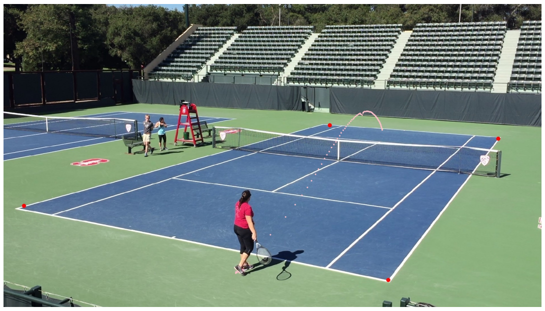


Fig. 16. Court and ball segmentation in video

In order to qualitatively evaluate our conversion from pixel to world coordinates, we create a two-dimensional plot of the path of the ball as it would be seen from a top-down view of the court and visually compare it to the path of the ball in the video. In Figure 17 plotted below, the sequential positions of the ball are marked with pink circles.

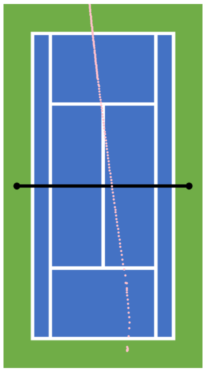


Fig. 17. Two-dimensional path of the ball

In addition, we create a three-dimensional scatterplot of the position of the ball in world coordinates. One circle is plotted for the position of the ball in each frame of the video, so the plot below in Figure 18 shows the path of the ball throughout an entire clip. As can be seen in the plot below, our algorithm produces a clear and reasonable path of the ball in three-dimensional world space.

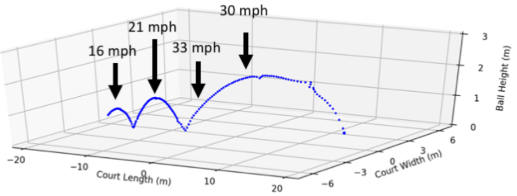


Fig. 18. Three-dimensional path of the ball

V. Discussion

A. Ball Segmentation Difficulties

One of our biggest challenges was consistently producing a noise-free ball pixel location. Despite trying and combining several image segmentation techniques, we still could not consistently produce a reasonably noise-free mask for the ball. Unfortunately, this step is fairly crucial in actually determining ball location. Ideally, if the ball is consistently identified in both frames as the only object of interest, then the ray intersection could be taken as a known measurement of the ball instead of unknown, and the Kalman filter performance would greatly be increased because measurements could be taken as known correspondences instead of unknown correspondences.

Within this step, one of our biggest challenges was that a moving player often produced several of the valid ball candidates. For example, the players hand moving to hit the tennis ball appears to follow ball-like trajectory and produces significant differences in the frame difference algorithm used to segment the ball.

B. Possible Improvements

As a future improvement, since videos are processed offline, we would consider several different adjustments to our algorithm. We would first consider player identification. This would enable us to avoid measurements of the ball within a certain proximity to the player. We would also compute a second localization pass after the first. Since the ball localization tracks closely but stray measurements cause deviations, a second measurement and filtering pass may smooth the trajectory. Instead of searching the whole frame for ball candidates, on the second pass we would instead search in a proximity of the estimated ball location for that frame. This would hopefully give us much more exact measurements for the true ball location at each frame time from which we could compute a more exact ball state.

Additionally, frame synchronization was also a challenge. For this project, we temporally synchronize the two videos clips by hand using visual cues in the clips. While the clips visually appear to be synchronized, automating this process would remove the potential for human error.

C. Limited Data Capture

Finally, due to our limited access to the tennis stadium, we were not able to capture stereo videos from additional cameras angles besides the one shown in this paper. It would be interesting to see how capturing videos from opposite ends of the court or from the sides of the court would impact the performance of our system. Capturing videos from additional camera angles could minimize occlusions of the line and/or tennis ball by the player, and perhaps would improve the depth estimation accuracy in our system.

VI. Conclusion

The goal of this project was to apply image processing techniques to stereo smartphone camera footage in order to bring ball trajectory analysis one step closer to recreational tennis audiences. Through the process described in this paper, we successfully identified court location and ball trajectories for stereo footage from our iPhone cameras. Although our system must be run offline and returns trajectories which are much less accurate than those of the Hawk-Eye system, the success of this project proves that tennis ball tracking is viable for the everyday tennis player.

Appendix

All team members were involved in all stages of the project, but each team member took the lead on the following tasks:

- Megan: Camera setup and calibration, video capture and editing, tennis player for data collection, visualization
- Tori: Ball detection, Kalman Filter
- Kyle: Court detection, multiple-view geometry, accuracy tests

It is additionally worth noting that Megan is enrolled in EE 367 and EE 368 and combined the two projects.

References

[1] T. Qazi, P. Mukherjee, S. Srivastava, B. Lall and N. R. Chauhan, ”Automated ball tracking in tennis videos,” 2015 Third International Conference on Image Information Processing (ICIIP), Waknaghat, 2015, pp. 236-240, `http://ieeexplore.ieee.org/document/7414772/`

[2] C. Lyu, Y. Liu, B. Li and H. Chen, ”Multi-feature based high-speed ball shape target tracking,” 2015 IEEE International Conference on Information and Automation, Lijiang, 2015, pp. 67-72. doi: 10.1109/ICInfA.2015.7279260., `http://ieeexplore.ieee.org.stanford.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=7279260&isnumber=7279248`

[3] Archana, M., Geetha, M.K.: An efficient ball and player detection in broadcast tennis video. In: Berretti, S., Thampi, S.M., Srivastava, P.R. (eds.). AISC, vol. 384, pp. 427436. Springer, Heidelberg (2016)., `https://link.springer.com/content/pdf/10.1007\%2F978-3-319-23036-8_37.pdf`

[4] Yan, F, Christmas, W and Kittler, J (2005) A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match In: British Machine Vision Conference, 2005-09-05 - 2005-09-08, Oxford, UK.

[5] B. Ekinci, M. Gokmen, ”A ball tracking system for offline tennis videos”, International Conference on Visualization Imaging and Simulation 2008, `http://www.wseas.us/e-library/conferences/2008/bucharest2/vis/vis06.pdf`

[6] Yu, X. Sim, C.-H. Wang, J.R. Cheong, L.F., A trajectory-based ball detection and tracking algorithm in broadcast tennis video, ICIP Image Processing, Vol. 2, pp. 1049- 1052, 2004, `http://ieeexplore.ieee.org/document/1419482/`