Producing a Tennis Shot Placement Report by Detecting Bounce Points in Tennis Shots

A dissertation submitted to the University of Dublin, in part fulfilment of the requirements for the degree of Master of Computer Science.

AUTHOR: CONOR GILMARTIN

SUPERVISOR: DR. KENNETH DAWSON-HOWE

SUBMITTED TO THE UNIVERSITY OF DUBLIN, TRINITY COLLEGE, MAY 2021.
ABSTRACT

Producing a Tennis Shot Placement Report by Detecting Bounce Points in Tennis Shots

BY

CONOR GILMARTIN

SUPERVISOR: DR. KENNETH DAWSON-HOWE

SCHOOL OF COMPUTER SCIENCE AND STATISTICS, TRINITY COLLEGE DUBLIN

This project presents a system that provides tennis players with a shot placement report. It does this by tracking the shots of the tennis player and detecting the bounce points of these shots. The intention is for this system to be used in a mobile application. The tennis shots therefore need to be tracked using a single mobile phone camera. The majority of systems used for tracking tennis balls make use of multiple high-speed cameras. There are difficulties in tracking tennis balls in videos due to the size of the ball as well as the speed it is moving at.

In this project, a review of the literature regarding previous techniques used to track tennis ball was carried out. It was found that previous solutions were similar in the steps that they did in order to track the tennis balls. However, it was also found that none of the systems had the intention of working on a mobile application. The design and implementation of the system for tracking tennis balls, detecting bounce points and generating a shot placement report is then discussed.

The system also needed to be tested and evaluated as a part of this project. Footage of tennis shots recorded with a mobile phone’s camera needed to be obtained. Ground truth for the bounce points in these videos also needed to be obtained manually. Finally, a testing framework was implemented in order to automatically test and evaluate the solution. Testing metrics were determined and were then calculated during this automatic evaluation process. 51% of the bounce points were detected by the solution, with an average of error of 9.97 pixels for the distance. Although not all of the bounce points were detected, the tracking of the tennis balls was generally successful, demonstrating that the system could be used in order to provide tennis players with a shot placement report.
DECLARATION

I, Conor Gilmartin, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signature:

Date: Monday 3rd May 2021
ACKNOWLEDGEMENTS

Firstly, I would like to thank my project supervisor Kenneth Dawson-Howe for his help and guidance throughout the year. His advice and expertise were invaluable during this project.

I would also like to thank the School of Computer Science and Statistics at Trinity College Dublin for the education throughout my undergraduate and Master’s Degree.

Finally, I would like to thank my family and friends for their support and encouragement during the year.
CONTENTS

1 Introduction .................................................................................................................. 8
  1.1 Project Goals ........................................................................................................ 8
  1.2 Motivation .............................................................................................................. 9
  1.3 Overview of Project ............................................................................................ 10
  1.4 Road Map ............................................................................................................ 11
2 Background ................................................................................................................ 12
  2.1 Tennis Hawk-eye System .................................................................................. 12
  2.2 Court Detection ................................................................................................... 15
    2.2.1 Overview ......................................................................................................... 15
    2.2.2 Line detection in Soccer Video ..................................................................... 16
  2.3 Tennis Ball Tracking and Bounce Detection .................................................. 20
    2.3.1 Overview ......................................................................................................... 20
    2.3.2 Steps ................................................................................................................ 20
    2.3.3 Ball Extraction ............................................................................................... 21
    2.3.4 Tennis Ball Candidate Classification .......................................................... 22
    2.3.5 Match Candidates to History ....................................................................... 24
    2.3.6 Player Tracking .............................................................................................. 25
    2.3.7 Bounce Detection .......................................................................................... 25
    2.3.8 Video Stabilization ......................................................................................... 26
    2.3.9 System Overview ........................................................................................... 28
3 Tracking Tennis Ball to Locate Bounce Points ................................................... 29
  3.1 Ball Detection ....................................................................................................... 29
    3.1.1 Motion Detection ........................................................................................... 29
    3.1.2 Colour Filtering ............................................................................................... 31
    3.1.3 Blob Detection ............................................................................................... 33
    3.1.4 RANSAC ......................................................................................................... 36
  3.2 Path Detection ...................................................................................................... 39
    3.2.1 Predicted Position ........................................................................................... 39
    3.2.2 Path Proximity .............................................................................................. 40
    3.2.3 Correcting errors in path .............................................................................. 41
  3.3 Bounce Detection ................................................................................................. 43
    3.3.1 Detect change of direction ........................................................................... 43
    3.3.2 Calculating bounce point .............................................................................. 44
  3.4 Path Elimination .................................................................................................. 45
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Shot Placement Report for Novak Djokovic [14]</td>
</tr>
<tr>
<td>1.2</td>
<td>Ball Flight Path Example</td>
</tr>
<tr>
<td>2.1</td>
<td>Hawk-Eye system using multiple cameras [15]</td>
</tr>
<tr>
<td>2.2</td>
<td>Hawk-Eye system diagram [1]</td>
</tr>
<tr>
<td>2.3</td>
<td>Court Detection system outline [3]</td>
</tr>
<tr>
<td>2.4</td>
<td>Zoomed-in frame from soccer video [2]</td>
</tr>
<tr>
<td>2.5</td>
<td>Example of gaps in white lines being filled in. [2]</td>
</tr>
<tr>
<td>2.6</td>
<td>Solution applied to one of the frames, showing the extracted white lines and the field colour [2]</td>
</tr>
<tr>
<td>2.7</td>
<td>Overview of System for Detecting Bounce Points in “Painless Tennis Ball tracking System” [3]</td>
</tr>
<tr>
<td>2.8</td>
<td>Tennis Ball extraction using yellow colour pane and thresholding [4]</td>
</tr>
<tr>
<td>2.9</td>
<td>Tennis Ball extraction using saliency detection and thresholding [4]</td>
</tr>
<tr>
<td>2.15</td>
<td>Shift in feature points [4]</td>
</tr>
<tr>
<td>3.1</td>
<td>The original frame</td>
</tr>
<tr>
<td>3.2</td>
<td>The absolute difference between the current frame and the background frame</td>
</tr>
<tr>
<td>3.3</td>
<td>Moving pixels in the frame</td>
</tr>
<tr>
<td>3.4</td>
<td>Yellow Pixels in the frame</td>
</tr>
<tr>
<td>3.5</td>
<td>The resultant binary image of the logical AND operation, showing the moving yellow pixels in the frame</td>
</tr>
<tr>
<td>3.6</td>
<td>Convex Blob on the left and Concave Blob on the right</td>
</tr>
<tr>
<td>3.7</td>
<td>Blobs with inertia starting at one and decreasing</td>
</tr>
<tr>
<td>3.8</td>
<td>The blobs detected in the binary image</td>
</tr>
<tr>
<td>3.9</td>
<td>The corresponding blobs detected in the original image</td>
</tr>
<tr>
<td>3.10</td>
<td>The image of the area around the blob</td>
</tr>
<tr>
<td>3.11</td>
<td>The edge pixels detected in the area around the blob found</td>
</tr>
<tr>
<td>3.12</td>
<td>Best circle found using the edge pixels found in the area around the blob found</td>
</tr>
<tr>
<td>3.13</td>
<td>Ball flight path with predicted position of the ball in the next frame</td>
</tr>
<tr>
<td>3.14</td>
<td>Ball flight path without size corrections</td>
</tr>
</tbody>
</table>
Figure 3.15 Ball flight path with size corrections using average of size found and predicted size ...........................................................................................................................................................................42
Figure 3.16 Bounce Detection with central point a part of the upward trajectory on the right and a part of the downward trajectory on the left...........................................................................................................................................................................44
Figure 3.17 Bounce Detection with points missing from downward or upward trajectory.....45
Figure 3.18 Tennis ball detected below previous point immediately after bounce point ......46
Figure 4.1 Image of tennis court .............................................................................................................................................................................................................................................................48
Figure 4.2 Mean Shift Segmentation applied to image of court ...........................................................................................................................................................................49
Figure 4.3 Canny Edge detection image obtained from result of Mean Shift Segmentation ..50
Figure 4.4 Contours found on tennis court .............................................................................................................................................................................................................................................................52
Figure 4.5 External contour found around tennis court .............................................................................................................................................................................................................................................................52
Figure 4.6 Diagram of tennis court for shot placement report .............................................................................................................................................................................................................................................................53
Figure 4.7 Original frame in video of tennis shots before transformation.............................55
Figure 4.8 The transformed frame using the perspective transformation that is used to transform points on court in the video to points on the court diagram ..................................................................................................................................................................................................................................................56
Figure 5.1 Successive frames around a bounce point where the ball is not touching the ground in either frame ..................................................................................................................................................................................................................................................60
Figure 5.2 Bounce Point where the tennis ball is touching the ground ..................................60
Figure 5.3 Simple Testing Metrics Table .........................................................................................61
Figure 5.4 Example frame from Test Video 3 ...............................................................................65
1 INTRODUCTION

1.1 PROJECT GOALS

The overall goal of this project was to create a mobile application that would provide tennis players with a report of their shot placement by tracking their shots. There were a number of steps that needed to be completed in order to achieve this goal.

The first objective that needed to be achieved was tracking a tennis ball through a video sequence from frame to frame. This included locating the tennis ball in each frame. It also included generating a ball flight path for the tennis ball over time. Given that the intention was for this to work as a part of a mobile application, the implementation needed to be effective for videos obtained from a mobile phone’s camera rather than a high-quality, high-speed camera.

Once the tennis balls were being tracked, the next aim was to determine when and where the tennis ball had bounced. This was done by analysing the ball flight path in order to detect where the shot landed on the court.

The last step in achieving the overall goal is producing a shot placement report with the locations of where the player’s shots landed. In order to provide the player with a clear visualisation of their shot distribution, the bounce points need to be shown on tennis court diagram rather than on the image of the tennis court from the video. It is therefore necessary to transform the bounce points detected. Consequently, the court needs to be located in the video in order to calculate this transformation.

The goal was to have all of these steps completed and working in real-time. This type of mobile application is a lot more useful if a tennis player can immediately see a report on their shot placement when they wish to see it.

Another objective of this project was to create a testing framework that would be used for evaluating the system developed for tracking tennis shots and detecting the bounce points. This required obtaining test footage of various types of tennis shots. It was also necessary to determine the ground truth for the bounce points in these test videos.
1.2 **Motivation**

Providing tennis players with a report showing the distribution of their shot placement is an extremely useful tool. It can aid them in visualising where their shots are landing.

Shot placement is important for a lot of tennis players and coaches. Normally, players will want their shots to be landing as close to the baseline as possible. The reason for this is because the return shot for their opponent will be more difficult if the ball lands closer to the baseline. It will result in their opponent’s return shot being hit from a greater distance.

It is also important for tennis players that their shots are both varied and close the sides of the court. The reason for this is because tennis players want their opponent to have to move a lot for each shot they have to play rather than stand in the middle of the court. Not only does it make the return shots more difficult for their opponents, the movement will also result in the opponent becoming more fatigued over the course of the match.

Showing tennis players a report of their shot distribution can help them improve their tennis shots. By displaying to them where their shots are landing, it can help them to understand how they need to change their shots in order improve the placement and distribution of their shots.

This type of information is important even at the top level of tennis for players like Novak Djokovic which can be seen in Figure 1.1.

![Figure 1.1 Shot Placement Report for Novak Djokovic](image-url)
1.3 **Overview of Project**

The first step in providing tennis players with a shot placement report is detecting the bounce points of their shots. In order to detect where the player’s shots are landing, a number of steps need to be taken. The first is that the tennis balls need to be tracked. In order to do this, the tennis balls are located in each frame. This is done by first locating the moving pixels and then locating the moving objects in the frame. It is then necessary to determine which of these moving objects are actually a tennis ball rather than another moving object. This can be done by using the size, shape and colour of the object.

Once the tennis ball is located in a frame, it then needs to be matched to a ball flight path. An example of a ball flight path can be seen in Figure 1.2 above. It is represented by the green circles in the image indicating the previous positions of the tennis ball in the video. If no paths exist, or if the ball does not match any of the existing paths, a new path is created. The balls are matched to the paths in a number of ways. One such way is by predicting the position of the next ball on the path and comparing this to the ball found. This predicted position is indicated by the white circle in Figure 1.2. Another way of matching the balls to the paths is by checking the proximity of the ball found to the location of the last ball on the path. Paths are deleted based on abnormal behaviour such as changing directions multiple
times or only moving a very short distance. They are also deleted when no there are no more tennis balls that match to them, meaning that the path has concluded.

Once the ball flight paths are obtained, the next step is to find the bounce points for the paths. This can be done by looking for a change in direction in the path. The bounce points that were detected in the above path are marked with a yellow cross in Figure 1.2. Once the bounce points are found, they need to be transformed onto a separate image. This will then act as the shot placement report for the player. This transformation is calculated by locating the line intersections of the court in the video and matching these points to the corresponding line intersection points in the separate image. These corresponding points are used to calculate a transformation matrix which is then used to transform the bounce points found in the video onto the shot placement report.

The court therefore needs to be located in order to find the position of the line intersections on the court. This only needs to be done once at the start as the court will not be moving in the video. It can be checked later to ensure that the camera has not moved.

Once the bounce points are detected for the tennis player’s shots, they can then be provided with a shot placement report showing the bounce points of all of their shots on the court diagram. They are provided with a clear visualisation of where their shots are landing on the court.

1.4 Road Map

This section will provide a brief overview of the structure of this thesis as well as the contents that will be discussed in the later chapters.

Chapter 2 will provide detailed analysis of the literature on tracking tennis shots, detecting bounce points and locating the tennis court.

Chapter 3 will discuss the methods used for tracking the tennis balls in a video sequence as well as detecting the bounce points of tennis shots. This will include details on locating the tennis balls in each frame, matching them to ball flight paths and locating where they bounced on the court.

Chapter 4 consists of a description of how the shot placement report is obtained from the bounce points. This will include details on how to detect the court and how to transform the bounce points onto a court diagram for the shot placement report.
Chapter 5 provides details of how the solution was tested. This includes how the testing videos were obtained and how ground truth for the bounce points was determined. This section will also include details of the testing framework developed. Finally, the results of the evaluation of the solution detailed in Chapter 3 and Chapter 4 will be provided and discussed.

Chapter 6 will conclude this report with a review of the system. Possible future work which could improve the application will also be discussed here.

2 BACKGROUND

In this chapter, a review of the literature regarding the tracking of tennis shots as well as tennis court detection will be provided. A research paper regarding the tennis Hawk-Eye system used for tracking tennis balls in order to determine where they have bounced will be discussed. Several papers related to locating the tennis court will also be discussed as well as research into tracking tennis balls through a video.

2.1 TENNIS HAWK-EYE SYSTEM

The most logical place to start when researching the tracking of tennis balls is with the Hawk-Eye system. It is the most well-known commercial system for tracking tennis shots. The system makes use of multiple high-speed cameras in order to track the tennis ball. The system is very different to the application proposed in this project which uses a single mobile phone camera. However, there are still many aspects of the system that are relevant to this project.

Originally, Hawk-Eye was a system developed for television broadcasters that could track the tennis ball during the match, providing data that could be used in virtual reality replays. It has since developed into a system used in professional tennis to determine where the ball bounced with very high accuracy in order to ascertain whether the shot landed inside or outside of the court. N Owens, C Harris and C Stennett [1] looked at this system outlining the design of the system as well as the challenges encountered.

It is essential for this system to operate close to real-time as the information gathered is only relevant for a short period of time. A virtual track of the previous shot is required within five seconds of the end of the rally. Without this efficiency, there would be too much of a delay for the system to be effective.
The Hawk-Eye system uses multiple cameras in order to locate and track the tennis ball as can be seen in Figure 2.1 above. Each camera had its own computer. Each individual camera performs 2D tracking separately. This was done in order to divide the computational labour within the system as well as reducing the communication between the separate camera systems. The results from the separate cameras are then combined elsewhere on a separate computer. For each frame, the ball is located. Two dimensional paths are built up over time on the image plane. These paths are then sent to the separate computer for 3D reconstruction. Once combined, the completed three dimensional path is then visualised in a virtual replay.

The initial Hawk-Eye system that was developed for television broadcasters made use of existing broadcast cameras. As a result, the system did not have control over when or where the camera moved or zoomed. Broadcast cameras are in a fixed location but they are going through continuous pan, tilt and zoom motion. This motion information needs to be extracted for each frame before any determined ball locations are useful.

Before tracking begins, the camera’s position is determined. This is done by tracking the court lines as they are visually prominent and their 3D location is known. The apparent width of the imaged lines can vary due to imaging effects. Therefore only the line centres are used. Straight lines are extracted by use of spatially adaptive thresholding, Local Mean Removal (LMR). Long segments are retained after straight line decomposition creating robustness against non-modelled image features like players. Tracking of the court may be lost at times, such as when the camera follows a high shot or when it zooms in on a player meaning many of the court lines are not visible. Reacquisition of the court’s location is done automatically.
Once the camera’s location and orientation are known, the system can then track the tennis balls. For each frame, it needs to locate the ball in the image. A tennis ball typically has a diameter of 2 to 10 pixels in this case depending on the location of the ball as well as the zoom. Normally, the tennis ball is moving quickly so it will appear as an ellipse or as sausage shaped streak due to motion blur.

LMR is used again for ball extraction. This method results in closed boundaries coming from the ball, the players, the court lines, and the net. These are then filtered based on size and shape resulting in the candidate balls. The centroids of the candidate balls are transformed into the coordinate system of a nominal stationary camera. This transformation is done using the information from the camera calibration and the pan, tilt, and zoom. This is necessary so that the movement of the camera does not affect the tracking of the ball.

The candidate tennis balls are matched to existing tracks. If they do not match any tracks, a new track is initiated. Once a track is initiated, a polynomial fit is used to predict the ball’s location in the next frames for that track. Over time as more points are added to the track, the polynomial fitting order increases. The radius of the acceptance region varies over time also, increasing depending on how long the matching for that path has failed. When tracking fails for too long an interval, the track is deemed complete. Valid ball tracks will be smooth and down-curving. The changes in directions that occur due to bounces or hits results in separate tracks. The completed tracks are assessed for length and movement and valid tracks are sent on for 3D reconstruction.

The system finds an approximate join point of two 3D tracks by using a quadratic model for each track and finding the intersection between them. This is used to initialise a Kalman filter, which takes in the 2D data directly. The system iteratively tries a fitting a linear, quadratic, cubic model of the incoming and outgoing compound tracks in order to get the best estimate of the impact point. A rule-based system is used to determine if the event was a strike, bounce or half-volley. The half-volley is difficult to detect as the upward bounce could be too short to track. A dummy track is inserted in this case. The bounce point is determined, and then outward velocity is estimated for this dummy track.
There are some aspects of the system that can be seen in Figure 2.2 above, that are not applicable to this project. This includes the use of multiple cameras meaning that 3D reconstruction is necessary as well as continuously tracking the court to determine the camera’s location in each frame. However, there are many aspects of the Hawk-Eye system that are relevant to this project. It is a system that both automatically locates the tennis court as well as tracking tennis balls. Although the ball is tracked in three dimensional space, it is first tracked on a single camera in two-dimensional space. Both of these features are applicable to this project. It is also a system that works in real-time which is important for this type of application.

2.2 COURT DETECTION

2.2.1 Overview

This section will provide an overview of the literature regarding detecting the location of the tennis court. W. Yoo, Z. Jones, H. Atsbaha and D. Wingfield [3] discuss the steps that are required in order to do this. This includes finding the lines in the image, finding the court from these lines, and then computing a transformation matrix in order to transform the points of the court in the image space into a separate ‘court’ space.
2.2.2 Line detection in Soccer Video

This paper [2] discusses how to detect the white lines on football pitch. This can therefore be applied to detecting the white lines on a tennis court. The first step in the process of locating the court is to detect the lines. Line detection includes three steps: field colours extraction, removal of unwanted frames and objects, and the detection of lines.

**Extraction of Field Colours**

The first step was to determine the dominant colour in the frame. This dominant colour would be taken as the colour of the field. In order to do this, a histogram was created for each of the red, green and blue colour channels. These histograms showed the amount of pixels in the frame that each value had ranging from 0 to 255. The highest occurring value from each of the three histograms was taken. Therefore, the most frequent red, green, and blue value was obtained from the image. These three values were taken to form the dominant colour in the image. This will be the colour of the field as it is seen in almost all of the frames.

**Removal of Unwanted Frames**

Only frames that show the soccer field were wanted. It is necessary to remove any frames that are useless in detecting the lines on the soccer pitch. These useless frames would include frames during commercial advertisements, frames that show the crowd, and frames that show the managers or the substitutes. Only frames that show the soccer pitch are useful in detecting the white lines, and these frames are extracted by using colour recognition. Frames that have more than half of their pixels within the field colour range will be extracted. Any others will be eliminated as they are determined to not be looking at enough of the soccer pitch.

Frames with zoomed-in, close-up views of the players on the pitch also have to be removed. These frames will often have a lot of pixels within the range of the field colour. They will therefore not be eliminated in the same manner. In order to eliminate these frames, the number of pixels within the field colour range in the central area of the image are counted. This is because the player will normally be in the centre of the frame in these zoomed-in frames. This can be seen in the image below in Figure 2.4. If the pixels counted does not exceed the threshold, it is eliminated.
Neither of these processes will be necessary in the application in this project as the camera will always be looking at the court. Therefore, all of the frames will be useful in detecting the lines on the court and for tracking the movement of the tennis balls.

**Detection of Lines**

Once the unwanted frames are removed, the white lines need to be detected. The lines are taken to be white in colour above a threshold. In this case, the threshold for the red, green, and blue values were 200, 210, and 190 respectively. Pixels that were below this threshold are replaced with field colour. This results in an image where only the white lines are present with all other pixels set to the field colour.

The white lines are then checked in order to patch together any broken lines. The eight neighbouring pixels of each non-white pixel are checked. If opposite pixels in any direction (horizontal, vertical, or diagonal) are white, the pixel is deemed to be a part of that line. It is replaced with a white pixel. This results in the white lines becoming continuous, fixing any gaps in the line. Example of the gaps in the white lines being filled in can be seen below in Figure 2.5.

![Figure 2.4 Zoomed-in frame from soccer video](image-url)
After the gaps in the white lines have been filled in, the resultant image contains the white lines, along with any other white features in the image. This could include player’s jerseys, sponsorships, or the ball among others. These other white objects need to be filtered out. In order to do this, the white pixels are scanned. Its four neighbours are checked, not including diagonal neighbours. If any of the four neighbours are white, that pixel is also checked. This process is repeated until the white pixel has no new neighbouring white pixels. The amount of white pixels that are connected is counted along the way. If there are too little white pixels
in a group, they are all replaced with the field colour. The lines are long and continuous. They will therefore have quite a high count of pixels in the group of white pixels. They should therefore not be filtered out by this method. The below images in Figure 2.6 shows the result of this method for a single frame. The solution successfully filters out the ball and the players wearing white jerseys. However, some of the lines are missed. This was due to the variance in the colour of the white lines. The white lines that were missed appear green in this image.

![Figure 2.6 Solution applied to one of the frames, showing the extracted white lines and the field colour [2]](image)

**Conclusion**

This method of line detection is not ideal for the tennis shot tracking application. Some of the features are not needed as the camera is not moving. Also, in an ideal scenario, the court will only be needed to be detected once as it should remain in the same location throughout as the camera should not move. There are still many features of this solution that are relevant to detecting the tennis court. However, it may not be the ideal solution as the tennis court will not be the only predominant feature in the frame, as is the case with the soccer field. The sky will be visible in almost all scenarios, meaning that the dominant colour may not be the court. Also, there could be many other white pixels in the image aside from the white lines of the court. Therefore, this method would not be the best solution for locating the tennis court in this application.
2.3 TENNIS BALL TRACKING AND BOUNCE DETECTION

2.3.1 Overview

In this section, three different studies focusing in the tracking of tennis shots will be discussed. [3, 4, 5] The main difficulties with tracking a tennis ball include the tennis ball blurring into background as it is moving so fast. The tennis ball is also subject to occlusion, false detection, and sudden change of motion detection. When the ball is close to a player, abrupt motion change can, and will likely, happen. This can happen at same time as occlusion and false detection.

"Painless Tennis Ball Tracking System" [3] provides an overview of the steps required for detecting tennis balls, tracking their flight paths, and detecting their bounce points.

"Automated ball tracking in tennis videos" [4] looked at tracking the tennis balls in high quality videos. It made use of both computer vision techniques as well as machine learning techniques in order to track the tennis balls.

“A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match” [5] looked at annotating a tennis match using a single low-quality camera. This was done by tracking the tennis balls in the video and detecting key events such as hits and bounces. A lot of tennis tracking applications make use of multiple, high quality video cameras. This system was developed for low-quality off-air video recorded with a single camera.

2.3.2 Steps

The various techniques used for tracking tennis balls and detecting where tennis shots land will be discussed in this section. In [3], they discussed the different steps needed in order to do this. W. Yoo, Z. Jones, H. Atsbaha and D. Wingfield developed their own system, the "Painless Tennis Ball Tracking System". However, the implementation of this system is not discussed in their paper. Instead, they provided an overview of the system and the various steps they implemented in order to track the tennis shots and detect the bounce points. These steps were similar in nature to those discussed in [4] and [5]. Although they varied in implementation at different stages, there were four basic steps to tracking the tennis balls and detecting bounces in a video. These were:

1. Motion Detection
2. Tennis Ball Candidate Detection
3. Match Candidates to History
4. Bounce Detection

These four steps can be seen in Figure 2.7 from [3].

![Figure 2.7 Overview of System for Detecting Bounce Points in “Painless Tennis Ball tracking System” [3]](image)

A review of the various implementations of these steps in the different systems will now be discussed. Player tracking is also discussed in this section.

### 2.3.3 Ball Extraction

In [5], pixel-wise temporal differencing used for motion detection. Before this happens, the image frames are de-interlaced into fields as a pre-processing step. Each frame has two fields with one field being every odd row in the frame beginning with the first row, and the other being every even row in the frame beginning with the second row. This was done because the tennis balls were moving so fast that they were alternately present and absent on successive frame lines. The difference between the field and several neighbouring fields is obtained. A threshold is then used to distinguish the moving pixels from the background.

In [4], the colour of the ball is used to extract the ball from the frame. The colour of the tennis ball is a bright yellow/green colour. An object of this colour appears white appears white in the yellow colour pane. Therefore, the frame is converted into the yellow colour pane in order to extract the ball. The following equation was used in order to do this:

\[
Y = g - \frac{r}{1.45} - \frac{b}{1.45}
\]

Y represents the yellow colour pane in this equation while r, g, and b indicates the red, green, and blue channels respectively. After the image is converted into the yellow colour pane, a threshold is used in order separate the tennis ball from the background. An example of this can be seen in Figure 2.8 below. The original frame is shown followed by the image
converted into the yellow colour pane. A threshold is then used in order to obtain the binary image.

![Figure 2.8 Tennis Ball extraction using yellow colour pane and thresholding [4]](image)

The saliency of features is also used in order to extract the ball from the image. Saliency features refers to the candidates of attention for human eyes in an image. It does not only make use of the colour of the pixels but also the motion in order to determine the salient features in the frame. Phase spectrum of Fourier transform is used to identify the salient areas in the video. In order to do this, each pixel in the frame is represented as a quaternion with these four channels: RG colour channel, BY colour channel, Intensity channel, Motion channel. The Phase Fourier Transform is computed for the pixel. When done for the entire image, this will indicate the most salient features in the frame. Again, a threshold is used to separate the ball from the background pixels. An example of this method can be seen in Figure 2.9 below. Again, the original frame is shown first. The second image represents the saliency calculated for each pixel before a threshold is used for the final binary image.

![Figure 2.9 Tennis Ball extraction using saliency detection and thresholding [4]](image)

### 2.3.4 Tennis Ball Candidate Classification

In [5], the moving foreground pixels from the motion segmentation from above are clustered into blobs. These blobs may be the tennis ball, but they may also be other moving objects. These objects could include a part of the player, the racket, or part of an advertisement board.
among various other objects. A method for determining whether these blobs are actual tennis balls or other moving objects is required.

Tennis balls have a standardized yellow colour. This can be used for classification. However, the colour of the tennis ball can be affected by the colour of the background it is travelling on. [6] The tennis ball is normally very small in the frame. Depending on the camera, the colour bandwidth could be very low, meaning that colour of the ball is greatly affected by the background. Therefore, more information is needed for classification.

Another method was proposed for classifying the candidate tennis balls. For each foreground blob, the area surrounding the blob is included. Bi-linear interpolation is used to obtain an enlarged intensity image of the blob. The corresponding binary image of the blob which was calculated earlier using a threshold on the moving pixels is also interpolated. The edge pixels of the binary image are extracted. These edge pixels are used as the edge of the blob in the intensity image.

An ellipse is fitted to the edge pixels. Least square criterion is used in order to do this. M points are sampled along the ellipse with the normal direction found for each point. The gradient is calculated using a 3 x 3 Sobel mask. An example of this method can be seen below in Figure 2.10.

![Figure 2.10 Edge detection for a moving ball [5]](image)

The mean absolute angle difference (\( \alpha \)) of the normal direction and the gradient direction at all sample points (\( \alpha_i \)) is used for blob classification. Blobs with a smaller \( \alpha \) are more likely to be actual tennis balls. The mean absolute angle difference is calculated using the following equation:
\[
a = \frac{1}{M} \sum_{i=1}^{M} a_i
\]

An eight-dimensional features vector is then created for blob classification. The mean absolute angle difference (\(a\)) is one of the features. Other features include the coordinates of the blob centre, the parameters of the fitted ellipse, and the mean of the pixels inside the blob in HSV channels. A support vector machine (SVM) is then trained to classify the blobs to find the actual tennis balls based on these features.

In [4], colour was already used to extract the tennis ball from the frame. They also analysed the blobs found in the frames as there were often more objects than just the moving tennis ball found. However, a much more simple solution was used in this scenario. The circularity of each blob is calculated. The blob with the highest circularity is then determined to be the tennis ball.

In this study, the images used had much clearer images of the tennis balls. Therefore, the ball was not elliptical in shape but was closer to a perfect circle. As a result, the tennis balls found had high circularities. The quality of the images also meant that the colour of the ball was more useful in this scenario as the colour of the ball was not affected greatly by the background colour. As a result, more complex methods for classifying the candidate tennis balls were not necessary.

### 2.3.5 Match Candidates to History

As mentioned previously, [3] does not provide any details of the implementation of the system. It therefore does not discuss how the candidate balls are matched to the history. It does however, mention that it fits a polynomial to the path in order to predict the next location of the ball. Furthermore, [4] also does not discuss matching the candidate balls to the history. The paper only deals with locating the tennis ball in each frame and does not track the ball from frame to frame.

Therefore [5] is the only paper that looks at the details of matching the ball to the history. This is done using a particle filter. There are two main steps of the particle filter. The first step is the prediction. It includes particle selection according to their weight resample. The
particles are resampled based on their weight, where they have a better chance of being selected with a higher weight. It also includes drift and diffuse. Each particle has a different state after applying drift, so predictions will be different. Noise is also added in order to cover unlikely scenarios. The second step is to update. This update, or correction occurs after the noisy measurement has been obtained. Each particle is evaluated and their weight is updated based on the calculated likelihood. [16]

2.3.6 Player Tracking
In [5], they also make use of player tracking in order to aid the tracking of the tennis balls. This is not necessary for tracking the tennis balls but it can provide further information that is useful when determining whether a shot has bounced or if it has been hit when there is a change in direction.

The tennis players are tracked using background subtraction with the absolute difference between the current frame and a background frame calculated. A simple blob tracker is then used in order to track the players.

The tracking of the players is not only used in order to determine whether the ball has bounced or has been hit. The players are also removed from the foreground image when detecting motion. Motion detection is used in order to locate candidate tennis balls. Since the players will most likely be moving, they are excluded from this search so that tennis balls can be isolated.

![Figure 2.11 Players tracked in tennis match [5]](image)

2.3.7 Bounce Detection
There was only one study [5] that discussed detecting the bounce points of the tennis shots. In [4], the focus of the study was just on tracking the tennis balls rather detecting any other events such as hits or bounces. As mentioned previously, [3] did not discuss any details of the implementations. It just outlined bounce detection as the last step in the process after tracking the tennis balls and their flight paths. Furthermore, this study only looked at practice shots
where there were no other players to hit the tennis ball back. Therefore, all changes in direction were determined to be bounce points.

In [5], the ball trajectories are tracked. Changes in direction in these trajectories are detected. These changes in directions are considered key events. These key events then need to be classified into bounces or hits. This is done by looking at a number of features of the key events. This includes the actual change in direction such as whether it changed direction in the x axis, or the y axis, or both. As previously mentioned, the position of the players is also taken into account when classifying the key events. A hidden Markov model is used to annotate the match based on the key events.

![Diagram of System for the Automatic Annotation of a Tennis Match](image)

**Figure 2.12** Diagram of System for the Automatic Annotation of a Tennis Match [5]

### 2.3.8 Video Stabilization

The video of the tennis match used in the study in [4] was obtained from a camera on a quadcopter. It therefore required stabilization before tracking the ball as the camera was exposed to jerks and noise. The steps involved in video stabilization can be seen in Figure 2.13 above. They are:

- Feature Detection and Matching
- Homography Estimation
- Parameter Smoothing
- Frame Warping

![Image](image1.png)

**Figure 2.14 Key Points detected by FAST algorithm [4]**

The FAST corner detection algorithm was used in this case. This algorithm finds key points in a time and memory efficient manner in order to compute the features. The key points in the above Figure 2.14 are shown in green. The corner key points are compared to the corresponding corner key points in the next frame. The shift of each key point due to unwanted motion in the video is calculated. The shift in the key points is shown below in Figure 2.15. The key points in the current frame are shown in green and the key points in the following frame are shown in red. The shift is shown with a yellow line.

![Image](image2.png)

**Figure 2.15 Shift in feature points [4]**
In the homography estimation, eight parameters are computed using the shift between the corresponding key points. The translation, rotation, scaling, skew, and perspective transformation is estimated.

The calculated transform is combined with the transformations for all camera motion after the first frame to obtain a cumulative transform with respect to the first frame. A filter is then applied to smooth these motion parameters.

Each frame is warped using the motion parameters obtained from this smoothing operation, resulting in a stabilized video.

2.3.9 System Overview

As previously mentioned, [3] gives an overview of the entire system. The end goal of the system that they outline is similar in nature to the goal of this application. Their system not only tracks the tennis shots and detects their bounce points. It also automatically detects the location of the court in the video. It uses this court location in order to calculate a transformation matrix in order to transform the bounce points detected from the image space into ‘court’ space. An overview of the system can be seen in Figure 2.16 below.

![Figure 2.16 Overview of Painless Tennis Ball Tracking System [3]](image)
In this chapter, I will discuss the design and implementation of the system developed for tracking tennis shots in order to detect the bounce points of those shots. Similar steps to those found in [3, 4, 5] were implemented in this solution. These steps include detecting the tennis balls and filtering them, detecting which path, if any, they are a part of, and determining the bounce points in those paths. The elimination of ball flight paths will also be discussed in this section.

### 3.1 Ball Detection

#### 3.1.1 Motion Detection

The first step in detecting the tennis ball in the frame is to locate the moving pixels in the frame. This was proposed in [5] where the current frame is subtracted from a background frame in order to find the moving pixels. The frame used as an example in this section can be seen above in Figure 3.1.

In order to detect the motion in each frame, the frame is first converted to grayscale. The next step is to get the absolute difference between this grayscale image and a background grayscale image. This background frame is initially set to the first frame of the video but it is updated once a ball flight path is complete. It is not updated during a path as it can interfere...
with the motion detection. Getting the absolute difference between these frames results in another grayscale image with larger values indicating large differences between the frames. The result of this action can be seen in the below greyscale image in Figure 3.2. A threshold is then used on this resultant grayscale image in order to obtain a binary image of the moving pixels in the frame.

![Figure 3.2 The absolute difference between the current frame and the background frame](image)

Once this was done, an opening operation was performed on this binary image. Opening is an erosion followed by a dilation. Erosion involves looking at all of the white pixels in the binary image, and changing them to black if they are not completely surrounded by other white pixels. Dilating the image involves changing any black pixels to white if they are touching any white pixels. Doing the erosion first removes any small regions of moving pixels that can occur due to noise. The dilation then needs to occur in order to preserve the size of the white regions in the image.

This opening operation was then followed with a closing operation on the resulting binary image. This is the opposite of the opening operation as it is a dilation followed by an erosion. This is done to combine any nearby white regions together as they are more than likely part of the same object. [12]
The resulting binary image showing the moving pixels in the frame can be seen above in Figure 3.3.

3.1.2 Colour Filtering

There were however often other moving objects in the frame besides the tennis balls. In order to filter out some of these objects, the colour of the tennis balls was used. This technique was used in [4] in order to extract the ball from the frame. In this case, the colour is being used to filter the moving pixels by colour.

The frame is first converted into hsv colour space. An upper and lower limit for the hue, saturation and lightness value are defined. This range encapsulates the colour of the yellow tennis balls. This range between the upper and lower limits needs to be large enough to accommodate the variations in the colour of the ball due to lighting and background changes in the video. The lower limit for the hue value is 25 whereas the upper limit is 35. This is the smallest difference between the limits as the hue value will not vary greatly compared to the other values. The saturation and lightness values have a lower limit of 45 and 100 respectively. The upper limit for these two values is the maximum value of 255. These are the values that can vary more greatly due to the conditions in the video and as a result, the range is larger. A binary image is then obtained using the inRange [17] function which returns whether each pixel falls within the defined range using the upper and lower limits.
The same opening and closing operations are performed on this binary image. Once again, this will eliminate any isolated pixels identified as yellow due to noise. It will also combine any closely identified yellow pixels as they are more than likely a part of the same object.

![Figure 3.4 Yellow Pixels in the frame](image)

The binary image above in Figure 3.4 shows the resultant binary image of the yellow pixels found in the original frame. This method correctly identifies the yellow balls. It also identifies some of the white lines of the court as yellow pixels. This occurs due to the wide range. It also occurs because the lines are white pixels where the hue value is not as relevant as the brightness is so high. It can therefore have the same hue value as the yellow pixels but with higher saturation and brightness values. The upper limit of these ranges still need to be high as either of the saturation or brightness values can be high due to conditions in the video. The white lines being identified as yellow pixels is not a major problem as these pixels will not be moving in the video.

Once this binary image has been obtained, it then needs to be used to find the moving yellow balls. A logical AND operation is performed between the binary image of the yellow pixels and the binary image of the moving pixels. This will result in a binary image indicating the moving pixels that are the same colour as the yellow tennis balls. The moving yellow pixels from the original frame in Figure 3.1 can be seen below in Figure 3.5.
3.1.3 Blob Detection

Once the binary image showing the moving yellow pixels is obtained, the next step is to find the moving yellow objects in this image. Blob detection [7] is used in order to do this.

Connected white pixels in the binary image are grouped together. The centres of the blobs are computed at this stage. [8] While this blob detection is done, certain features of the blobs are calculated. These can be used to filter out some of the objects found.

The size and the circularity of the blob were used in order to eliminate certain objects. A minimum and maximum size as well as a minimum circularity for the blobs were set. If any of the blobs found were not within these limits, they were disregarded. The range for the filter for the size of the ball had to be quite large as the size of the ball could vary a lot based on how close it was to the camera. There was no maximum value set for the circularity of the objects. Circularity is defined by the equation,

\[
\frac{4\pi \text{Area}}{(\text{perimeter})^2}
\]

A perfect circle would have a circularity of one meaning that this is the de facto maximum value for the objects. The tennis balls found in the videos normally had a high circularity.

The blobs found can also be filtered by convexity at this stage. A blob is defined to be convex if it contains the line segment that joins any two points of the blob. Otherwise the blob is
Convexity refers to the measure of how close a concave blob is to being convex. It is defined as the area of the blob divided by the area of its convex hull. A convex hull is the smallest convex figure that surrounds the entire blob. If there are any concaves in the blob, the convexity will be decreased. A convex blob, such as a perfect circle, would have a convexity of one. In Figure 3.6 below, the convexity of the blob on the left will be higher than the blob on the right.

Convexity was not used in this application to filter the blobs as the convexity of the candidate balls found were not consistent. The entirety of the ball was not always found so it was not certain what shape the blobs of the candidate balls would be.

Inertia could also be used at this stage to filter the blobs. Inertia is essentially a measure of how elongated a blob is. A perfect circle would have an inertia value of one whereas a line would have an inertia value of zero. An ellipse would have an inertia value between zero and one depending on how long it is compared to its width. [8]

Once again, this feature was not used to filter the blobs. This is because, depending on the speed of the shot, the shape of the ellipse would be different. For slower shots, the ball found would be a circle. For faster shots, the ball found would be an ellipse due to motion blur. [1] As a result of this variation, this feature of the blobs found could not be used to filter any of them.
The image in Figure 3.8 below shows the blobs found in the binary image. Neither of the two blobs were filtered out in this case due to size or circularity. These blobs correspond to the ball and the hat in the original frame. This is shown in the second image in Figure 3.9. The hat is found in this scenario due to the fact that it is the same colour of the tennis balls and it is moving in this frame. It is also similar in size to a tennis ball. This is not a problem as the hat will not be moving in a manner that is normal for a tennis ball that has been hit. It will not move a lot from frame to frame and it will most likely not move in a consistent fashion. When detecting the ball flight paths, it can therefore be eliminated due to its abnormal movement, as it will not be moving in a manner similar to a tennis ball in motion.

Figure 3.8 The blobs detected in the binary image
3.1.4 RANSAC

In order to achieve a higher accuracy in the location of the tennis balls in each frame, I did further processing at the location of the blobs detected. I looked at the area around the candidate balls detected.

Firstly, edge detection is done in this area. The canny algorithm is used in order to do this. An example of this can be seen in the image below. The canny edge detection algorithm makes use of thresholds and therefore returns a binary image as output. It analyses the greyscale image to see if there are any sudden changes in brightness. It uses two thresholds. The change in brightness is compared to these thresholds. If the change is higher than the high threshold a strong edge is detected. If the change is not higher than the high threshold but is higher than the low threshold a weak edge is detected. It then removes any weak edge points that are not connected to a strong edge point. [10] The high and low thresholds had to be altered so that it...
detected the edges of the tennis ball but did not detect too many other edges as this would affect the ability to locate the ball. The low threshold in this scenario was 150 while the high threshold was 450.

A RANSAC method was then implemented in order to find the best circle that fit the edges found. This method involves repeating the following steps. It first picked three edge points at random. This is done by adding all of the edge points in the image into a list. Three random numbers that are less than the size of this list are then chosen and the edge points at these indices are used as the three random edge points.

The centre of the circle that contained these three points is then calculated as well as the radius of the circle. The equation of a circle is:

$$(x - h)^2 + (y - k)^2 = r^2$$

By substituting the x and y values from the three different points into this equation, three different equations will be formed. These three equations have three unknown values: h, k, and r. The equations are then solved to get the centre of the circle, (h, k), as well as the radius, r.

It then checked to see how many other edge points were also on this circle. This was done by looping through different points on the circle and checking to see if they were any edge points at those locations. The points on the circle were looped through by incrementing the angle and then finding the point of the circle based on the radius and this angle. When checking to see if the points on the circle were edge points, a region of acceptance was used.
so that nearby edge points were considered as a part of the circle. The radius of this region of acceptance depended on the radius of the circle. The larger the circle was, the larger the region of acceptance was. This process was repeated a number of times in order to find the optimal solution. In this solution, the process was repeated 100 times in order to ensure that the best circle was found in the image.

The optimal solution was determined by the amount of edge points lying on the circle found. There was a minimum radius for the circles found in order to eliminate small circles that occur throughout the image. There was a minimum amount of edge points that had to be lying on a circle in order for it to be accepted as the best circle. If the best circle found by the solution did not reach this minimum value, it was determined that no circle was found in the region. [11]

The location and size of the candidate ball were updated with those of the circle found. If no circle was found, the location and size of the candidate ball remained the same. Even if no ball is found with this method, the blob could still represent a tennis ball in the frame. Therefore, the blob is not eliminated if no circle is found using the RANSAC method. The size and location of the blob detected is still used for the size and location of the candidate ball.

![Figure 3.12 Best circle found using the edge pixels found in the area around the blob found](image)

There were some issues with this method. Sometimes, the canny edge detection algorithm would detect an edge where there is a shadow on the ball. This was especially applicable when the ball was closer to the camera as it was more visible. The ball was also occasionally similar in colour to the background. When this occurred, the difference between the ball and the background would therefore not be distinguishable enough in order for an edge to be
detected. A lot of the time, these two issues were not significant as the RANSAC method would still find the circle around the ball if enough of the edges surrounding the outside of the ball were detected. This can be seen in the above example using the image in Figure 3.10. The binary edge image was obtained from this image using canny and can be seen in Figure 3.11. It was determined that the shadow on the ball was an edge through the middle of the ball. It also did not detect an edge at the bottom right corner of the ball. In spite of this, the tennis ball was still located by the RANSAC implementation. The image in Figure 3.12 displays the circle found with the RANSAC method using the binary edge image. The circle found still represents the tennis ball in the image.

3.2 PATH DETECTION

After the candidate tennis balls are found in the frame, the next step is to determine whether or not they are a part of any existing paths. If they are not determined to be a part of any existing ball flight path, a new path is created with that ball as the starting point. There are a few different methods used in order to see whether or not the ball was a part of an existing ball flight path.

3.2.1 Predicted Position

For each existing path, the position of the tennis ball in the next frame was predicted. This prediction was based on just the previous two balls in the path. This is sufficient in this case as the ball is not moving a lot in each frame. Over this short difference, the curve of the path is not noticeable. It is moving in a linear-like fashion. The predicted position is obtained by getting the difference between the two previous points. This difference is then added onto the last position in the path. The predicted position for the next frame is shown in the image in Figure 3.13 with the white circle. More complex solutions could be used for predicting the position of the tennis ball in the next frame for a given path. For example, the flight of the ball will closely resemble that of a quadratic curve. Three points could be used in order to fit a quadratic curve to the previous points. Also, more points could be used in order to find a more accurate quadratic curve using regression. However, these methods are more computationally expensive than just using the previous two points. The prediction for the position of the ball does not need to be extremely accurate as it is only used to check whether candidate balls are a part of this path. They are also used to narrow the search area when no candidate balls are matched to a path.
The candidate balls are compared with this prediction. If they are within a certain tolerance to the prediction, it is determined that the ball is a part of that path. The tolerance in this scenario was ten pixels. This tolerance was adjusted to accommodate for errors in the predictions but to not include candidate balls that were not part of the ball flight path.

3.2.2 Path Proximity

There will be instances when the ball does not match the predicted position for the path but is still a part of that path. This will occur at the very start of the path when there are not enough points for a prediction. It will also occur when the ball changes direction. The Hawk-Eye system is designed so that the path ends when the ball changes direction. It starts a new path in these scenarios. However, this was not the best approach in this application. The Hawk-Eye system found bounce points by finding the intersection of different ball flight paths. However in this application, there were often other moving yellow objects in the video. It would therefore be more difficult to determine which intersections of flight paths were bounce points, and which were two completely separate paths. It was therefore more applicable in this application to continue the flight path when it changed direction and determine the bounce points in another way.
For this reason, if no candidate balls match the prediction of the path, the candidate balls are then checked to see how close they are in proximity to the last point on the path. The distance is checked to see if it is below a certain tolerance. In this case, the distance was 120 pixels. This had to be large enough to include all of the tennis balls when they were moving at high speed. At this stage, it was also ensured that the distance between the two balls was not too small. The tennis balls were always moving from frame to frame. Ensuring that the tennis balls found have moved more than this minimum distance eliminated slowly moving objects such as the hat in this scenario. The minimum distance was set to five pixels. The minimum distance still had to be relatively short because at times, if the ball was moving towards the camera, the distance that it moved in a single frame was quite small.

At this stage, all of the candidate balls have been checked to see if they are a part of existing paths. Both of the prediction for the path and the proximity to the last point on the path were used in order to match candidate tennis balls to the ball flight paths. Therefore, any candidate balls that were not a part of any of the existing paths are used to form the start of a new path. The position and size of the candidate ball is used as the starting point of the new path.

3.2.3 Correcting errors in path

If the candidate ball is matched to an existing ball flight path, some corrections are made to the path. The prediction for each path comprises both location and size. It is not likely that the size of the tennis ball is going to vary a lot from frame to frame. As a result, an average of the predicted size of the ball and the actual size of the ball found is calculated. The size of the ball is set to this average size.

There can be errors in the size of the ball found. There are various reasons for this. It can occur due to lighting conditions such as some of the ball being shaded and therefore difficult to find accurately. It can also occur when the background is similar in colour to the colour of the tennis ball. This correction ensures that when these errors occur, it does not affect the ball flight path.
The first image above in Figure 3.14 shows the path without any corrections while the second image above in Figure 3.15 uses the average of the predicted size and the size found for the tennis balls in the flight path. The ball flight path in the second image is smoother and more consistent. It is therefore more representative of the tennis ball’s actual ball flight path.
There are no corrections made for the position of the ball based on the prediction. This is because it is more difficult to predict the position of the ball rather than the size of the ball. Therefore, inaccuracies introduced in the predictions of the locations of the ball would affect the positions of the tennis balls in the path. The predictions are accurate enough to check to see whether a candidate tennis ball is part of a flight path and to narrow down the search radius, but it is not accurate enough for it to be used to affect the actual position of the tennis ball found.

3.3 **BOUNCE DETECTION**

3.3.1 **Detect change of direction**

After detecting the tennis balls in the frame and adding them to ball flight paths, the next step in the process is to detect bounce points in the paths. Similar to [3], this application assumes that it is detecting tennis shots in a practice scenario. It therefore, determines that changes in directions in the tennis ball flight paths are bounces, rather than another player hitting the ball.

The first thing that needs to be done is to detect upward changes in direction. In order to do this, the y positions of the balls in the path are checked for each new ball added to the path. The solution keeps track of whether the tennis ball is moving in an upward or downward direction at all times. This is done by checking whether or not the y value of the latest ball found is higher or lower than that of the previous ball in the path. While the tennis ball is moving downwards in a path, the solution is searching for bounce points. If the latest ball is found to be higher than the previous ball, this is regarded as an upward change of direction and therefore a bounce. In the diagram shown below in Figure 3.16, the bounce would be detected at the fourth ball in this sequence. The bounce in reality occurred before this point but this is the first point detected that is higher than the previous point.
3.3.2 Calculating bounce point

Once a bounce is detected, the bounce point can then be calculated after the next frame. The bounce point cannot be detected immediately as at least two points from the upward trajectory is needed. The point immediately previous to the point where the bounce point was detected cannot be used in this calculation. In the diagram above, this is the third point in the sequence. It is not certain whether this point occurred before or after the actual bounce point. It could therefore be a part of either the downward or upward trajectory. Two points are needed from both the downward and upward trajectory in order to find the bounce point. As a result, it is necessary to wait until after the next point is found in order to calculate the bounce point. Once this is done, the last two points can be used to form one line as they are a part of the upward trajectory. The next step is to use two points from the downward trajectory to form the second line for this calculation. As stated above, the point immediately previous to where the bounce point was detected cannot be used as it is not known whether this point was a part of the downward trajectory. However, it is certain that the two points previous to this point were a part of the downward trajectory, so they can be used to form the second line. The intersection of these two lines is calculated in order to find the bounce point.

Using straight lines in this case is once again sufficient. As previously mentioned, there is not much curve to the ball flight paths in this scenario as the points are so close together due to the ball not moving much from frame to frame. It was therefore not necessary to use more of the points in order to form a quadratic curve to fit the ball flight path.
There are a few different scenarios where this calculation for the bounce point is not possible. This will occur when there are not enough points around the bounce point where tennis balls were detected. This can occur if there is a bounce point at the very start of a ball flight path. There may not be two tennis ball points that are a part of the downward trajectory in this case. Similarly, this scenario could occur for the upward trajectory. If the ball moves out of frame following a bounce point, there may not be two tennis ball points that are a part of the upward trajectory. In both of these scenarios, two lines cannot be formed in order to find the intersection between them. In this case, the point immediately previous to where the bounce point was detected is determined to be the bounce point. This method of determining the bounce point is not as accurate but without more points in the path, it is the only solution.

![Figure 3.17 Bounce Detection with points missing from downward or upward trajectory](image)

### 3.4 Path Elimination

There are a couple of different scenarios where a ball flight path will be eliminated.

#### 3.4.1 Abnormal Movement

The ball flight paths are analysed to see if they are behaving like a tennis ball that has been hit would. A tennis ball’s flight path will be consistent. There will not be many changes in direction. Abnormal direction changes are detected in the ball’s flight path. When they are detected, the path is deleted. For example, if the ball immediately moves back downwards following a bounce point, the path is deleted. An example of this can be seen in the diagram below in Figure 3.18. This also occurs when the ball changes direction multiple times in quick succession along the x axis.
3.4.2 Ball not detected

If no ball is detected for a path, one last attempt is made to locate a ball for that path. The predicted position of the ball for the path is used to narrow down the search area. The area around the predicted position is checked.

Earlier in the process, once the blobs were found for the candidate balls, a method was used to obtain a more accurate location and size of these candidate tennis balls. This method involved edge detection followed by a RANSAC method in order to find the best circle in the area around the blob found. It was described earlier in this report. This same edge detection and RANSAC method is used at this point in order to try and locate the ball. If no blob was detected in this region, it is still possible that the ball was there and this method hopes to find it. Up until this point, tennis balls have only been located using the blobs detected. It is still possible that this method of detecting the edges and finding a circle can locate a ball for the path in the predicted region even if no blob was found in this area.

If a ball is found in this fashion, the ball is added to the path. The path is not deleted and the solution will move on to the next frame. If no ball is detected for a number of frames, it is determined that the path has finished, and is deleted.

3.4.3 Missing Frames

As mentioned above, a path is deleted if no ball is matched to it for a few frames. However, the flight path is not deleted if the ball is not detected in just one frame. A counter is stored
for each of the flight paths indicating the amount of frames that no tennis ball has been detected for that path. If no ball is found with any of the methods mentioned above, this counter is incremented. It is reset to zero if a tennis ball is found for that path in any frame. When no ball is found, the counter is also checked to see how many consecutive frames have passed without any balls being matched to it. If it is above a certain number, in this case two, it is determined that the path has finished, and the path is therefore deleted.

### 3.4.4 Path Deletion

There are a number of reasons why paths are deleted. It can occur when the path is determined to not actually be the flight path of an actual tennis ball. It can also occur when the no tennis balls are matched to a ball flight path for a number of frames. This can occur when in different scenarios, such as when the ball moves off screen or if the ball is not detected in the frames. In all cases, the path is deleted.

When this deletion occurs, the amount of points in the path are checked. If the amount of points in the path exceeds a minimum value, it is concluded that the path was indeed an actual ball flight path. The minimum value in this application was ten points. In cases where the size of the flight path being deleted exceeded this minimum number of points, and was thus determined to be an actual ball flight path. When this occurred the first bounce point from the flight path was added to the list of the final bounce points. Only the first bounce point is added as this is the only bounce point that tennis players are interested in. This is because any further bounces are not in play in a tennis match. Any bounce points detected in paths that were determined to not be actual tennis ball flight paths, were not added to the final bounce point report.

### 4 Shot Placement Report

In this chapter, I will discuss the system used for displaying the bounce points found to the player in a shot placement report. It first involves locating the court in the image. The points of the court are then used to calculate a transformation matrix used for transforming the bounce points onto the court diagram.
4.1 COURT DETECTION

![Image of tennis court](image)

*Figure 4.1 Image of tennis court*

4.1.1 Mean Shift Segmentation

There were two methods that I have tried implementing for locating the court automatically. The first method that I looked at for locating the court was by using mean shift segmentation.

Mean shift segmentation will cluster pixels together. It will do this based on both the proximity of the pixels to each other and their colour. It therefore provides both spatial and colour segmentation in the image. Mean shift segmentation does not require the number of clusters prior to execution. It will determine the amount of clusters based on the features and regions in the image.

In this algorithm, for each pixel, the local kernel density is estimated as well as the direction of the local increasing density, which is the mean shift vector. The kernel smooths all of the data sample, and adds them together. A spatial kernel and a colour kernel are used, which limits the pixels included in the estimation of the kernel density to points that are both close in proximity and colour to the current pixel. The kernel width has to be selected for both the spatial and colour kernels.

The spatial kernel width determines the area around the current pixel that is included in the estimation. This was defined as having a radius of 30 pixels in this case. The areas that the application is searching for is quite large, meaning that this value could be larger as small
regions were not relevant to the application. However, it could not be too large as it was important that the separate regions of the court were separated by the white lines. A lot of the time the area outside of the court lines is the same colour as the court. Therefore, this area needs to be separated into a separate cluster than the court due to the white lines.

The colour kernel determines the pixels to be included in the estimation in the area defined by the spatial kernel. Only pixels similar in colour to the current pixel are included in the mean. Once the estimation for the local kernel estimation and the mean shift vector are determined, the pixel is shifted to the new mean. This process is repeated until the location stabilizes.

All pixels whose location stabilizes at the same point are identified as a part of the same cluster. The mean of these pixels of the pixels grouped together into the cluster is calculated and applied to all of the pixels in the cluster.

As you can see in the image below in Figure 4.2, it segmented the different regions of the court nicely in this example. A downside to using this method is that it is quite time consuming. However, in this application, the court only needs to be found once as the camera is not moving. Therefore, the efficiency of the algorithm used is not critical.

![Figure 4.2 Mean Shift Segmentation applied to image of court](image)

The problem with this solution was that only the image is altered. Information about the regions such as their location in the image was not obtained by using this solution. As a
result, this solution could be used to alter the image, but another implementation was needed in order to actually locate the court in the image and to locate the points for the line intersections of the court.

Although mean shift segmentation was used in this case to directly locate the court, the resultant image of the region segmentations could still be used in order to aid in the location of the court. [20]

4.1.2 Contour Detection

The second method used in order to locate the court involved using contours. The resultant image from the mean shift segmentation was used in this method rather than the original image of the court. This is because a lot of the noise has been filtered out in this image. The edges are much more prominent in this image as a result.

![Figure 4.3 Canny Edge detection image obtained from result of Mean Shift Segmentation](image)

In this method, an edge image is first obtained using the canny edge detection method. This returns a binary image showing the edge points. The result of this can be seen in Figure 4.3 above.
The next step is to find all of the contours within the image from the edges found. [18] In order to do this, the edge points are connected together, forming contours. This method looks at the edge image to find connected edges (i.e. contours). The edge image is represented by a graph. The nodes represent the edge pixels with a cost and orientation for each pixel. Nodes are connected if the pixels are adjacent to each other and if they have the same orientation. Once all of the adjacent edge pixels are connected into edge chains, the chains are then modified to fill in small gaps. It also removes duplicate contours that are very close to each other.

While detecting these contours a lot of them can be filtered out. The contours can be filtered based on size because the contours that are formed on the court lines are quite big compared to others. They can also be filtered by the amount of corners the contours have as the areas in the court we are trying to locate are all rectangular. [19]

One problem with this method is that the net can obstruct some of the areas that are being looked at and therefore cause the contour to be in the wrong spot, or have too many corners, in which case, it would be filtered out. Another problem is that the contours do not always go to the actual corner of the section being looked at so it will not be completely accurate. Another problem is that the entire court is usually not visible to the camera. As a result, the contour surrounding the entire court cannot be found like the second image shown. Even in this case, the actual corners still need to be determined as there are more than four corners in the contour due to the obstruction of the net.

Despite these issues, the contours were found successfully in the images. As you can see in Figure 4.4 below, the contours were successfully found for the different regions of the tennis court. Similarly in Figure 4.5 below, the exterior contour surrounding the entire court is found successfully.

Once the contours are found, their positions and size can be used to determine which section of the court it is. The corner points of the contours can then be used as the points for the intersection of lines on the court.
Unfortunately, these methods were not implemented completely as a part of this project. In the case of locating the contours of the individual regions of the tennis court, the contours found could not be matched correctly to the different areas of the court. It could not be determined which contour matched to which region of the tennis court. Therefore the line intersections could not be determined. In the case of locating the exterior contour of the entire
court, there were two problems. As previously mentioned, the contour could be obstructed by other objects, meaning that it was difficult to determine which points on the contour corresponded to the corner points of the tennis court. A bigger problem was that the entire court was not visible in most instances, so the exterior contour surrounding the entire court could not be found. The points of the court were located manually so that the transformation could still be completed.

4.2 BALL PLACEMENT REPORT

4.2.1 Geometric Transformation

In order to provide the players with a shot replacement report, the bounce points needed to be transformed onto a separate image. This image is a diagram of a tennis court. It can be seen below in Figure 4.6. This court diagram is used for the shot placement report instead of an image of the court as it provides a clearer visualisation of the shot placement distribution for the player.

![Figure 4.6 Diagram of tennis court for shot placement report](image)

The positions of the intersections of the lines of the court have been found. The corresponding points of the line intersections on the court diagram in the separate image are known as this image does not change. A geometric transformation is calculated using the points from the court in the video and the equivalent points in the image of the court diagram.

Only points from the near side of the court are found and used in this geometric transformation, as it is not viable for a mobile phone camera to look at both sides of the court due to the quality of the image.
A Perspective transformation is used. The court is a planar surface that is not parallel to the image plane. It can therefore not be corrected by using an affine transformation.

We therefore need a perspective transformation that will transform a point \((i, j)\) in the image space to the point \((i', j')\) in the transformed space. A transformation matrix is calculated in order to do this. The perspective transformation using the matrix will look as follows:

\[
\begin{bmatrix}
i \\
j \\
w
\end{bmatrix}
= \begin{bmatrix}
p_{00} & p_{01} & p_{02} \\
p_{10} & p_{11} & p_{12} \\
p_{20} & p_{21} & 1
\end{bmatrix}
\begin{bmatrix}
i' \\
j' \\
1
\end{bmatrix}
\]

The unknown variables \(p_{xy}\) in the transformation matrix need to be calculated. From matrix multiplication rules, we know the following three equations:

\[
i.w = p_{00}.i' + p_{01}.j' + p_{02}
\]
\[
j.w = p_{10}.i' + p_{11}.j' + p_{12}
\]
\[
w = p_{20}.i' + p_{21}.j' + 1
\]

The equations for both \(i\) and \(j\) can therefore be inferred:

\[
i = p_{00}.i' + p_{01}.j' + p_{02} - p_{20}.i'.i' - p_{21}.i'.j'
\]
\[
j = p_{10}.i' + p_{11}.j' + p_{12} - p_{20}.j'.i' - p_{21}.j'.j'
\]

At least four corresponding points are needed in order to solve for the unknown variables in the transformation matrix. Each of the four corresponding points results in an equation for both \(i\) and \(j\). There are therefore eight equations. There are eight unknown variables in the transformation matrix. The eight equations can then be solved simultaneously in order to determine the unknown variables. This simultaneous equation can be simplified with the following matrix multiplication:
In order to solve for the unknown variables \((p_{00} - p_{21})\), the inverse of the square matrix is calculated. This is then multiplied by the first vector containing the observed points from the original image.

Once the perspective transformation matrix is calculated, it can then be used to transform points in the original image space into the transformed, court diagram image space. An example of the transformation applied to one of the frames in the video is shown below in Figure 4.8. The original image is shown in Figure 4.7. Only points from the near side of the court are used for the perspective transformation. As a result, only points located within this region are clearly visible in the transformed image. Any points located outside of this area within the original image are distorted in the transformed image. This is not a problem however, as this application is only searching for bounce points in this region.

Figure 4.7 Original frame in video of tennis shots before transformation

\[
\begin{bmatrix}
i_1' & j_1' & 1 & 0 & 0 & 0 & -i_1i_1' & -i_1j_1' \\
0 & 0 & 0 & i_1' & j_1' & 1 & -j_1i_1' & -j_1j_1' \\
i_2' & j_2' & 1 & 0 & 0 & 0 & -i_2i_2' & -i_2j_2' \\
0 & 0 & 0 & i_2' & j_2' & 1 & -j_2i_2' & -j_2j_2' \\
i_3' & j_3' & 1 & 0 & 0 & 0 & -i_3i_3' & -i_3j_3' \\
0 & 0 & 0 & i_3' & j_3' & 1 & -j_3i_3' & -j_3j_3' \\
i_4' & j_4' & 1 & 0 & 0 & 0 & -i_4i_4' & -i_4j_4' \\
0 & 0 & 0 & i_4' & j_4' & 1 & -j_4i_4' & -j_4j_4' \\
\end{bmatrix} \cdot \begin{bmatrix}
p_{00} \\
p_{01} \\
p_{02} \\
p_{10} \\
p_{11} \\
p_{12} \\
p_{20} \\
p_{21} \\
\end{bmatrix} = \begin{bmatrix}
p_0 \\
p_1 \\
p_2 \\
p_3 \\
p_4 \\
\end{bmatrix}
\]
4.2.2 Shot Location Display

Once the bounce points are calculated, these points are transformed from the image space into the court diagram image space using the perspective transformation matrix calculated. The transformed points are drawn onto the court diagram. This image of the court diagram with the player’s bounce points marked can then be shown to the player when they wish to view. It will display to them a report on their shot placement.

An example of the bounce points being drawn onto the court diagram is shown below in Figure 4.10. The bounce points drawn on the diagram correspond to the bounce points detected in Figure 4.9. The image shows both bounce points being drawn on the diagram. However, when the path is completed, only the first bounce point will be stored and drawn onto the court diagram for the shot placement report.
Figure 4.9 Ball flight path with bounce points marked

Figure 4.10 Bounce points from the path in Figure 4.9 marked on the court diagram
5 EVALUATION

This chapter will discuss how the system developed was evaluated. The testing process for the detection of bounce points will be discussed first. The steps taken to test the location of the court will also be discussed.

5.1 BOUNCE LOCATION

In order to evaluate the system being implemented, a system needed to be put in place in order to test it. There were a number of steps involved in developing this testing system. Firstly, footage of tennis shots needed to be obtained as well as the ground truth for the bounce points in these videos. Then, suitable testing metrics needed to be decided in order to evaluate the performance of the solution. Finally, the testing system needed to be implemented so that the solution would be automatically evaluated using all of the test videos and the ground truth obtained, and calculating the testing metrics for each video.

5.1.1 Test Videos

In order to test the solution on the detection of bounce points, footage of various tennis shots needed to be obtained. The videos obtained were recorded as a precaution before the last lockdown. No more test videos have been able to be recorded since then as all tennis clubs were closed due to COVID restrictions.

Various shot types were included in these recordings. There were videos showing both slow and fast forehands, serves, as well as footage of games between two players. Only videos containing footage of tennis shots hit in a practice environment were used for testing purposes. No videos of tennis matches played between two players were used when testing the solution.

The camera used for recording the videos used for testing was attached to a fence beside the tennis court. When the ball struck this fence, the entire fence was shaken. The camera was therefore also shaken. This motion in the camera caused difficulties when determining which pixels were moving in each frame.

There were nine videos in total which had an average time of two minutes and 54 seconds. However, only four of these videos were used for testing as the other five involved two
players playing a tennis match. The first two videos contained footage of multiple forehands. The shots in the first video were moving at a slower pace while the shots in the second video were hit with more power. The quicker pace of these shots was more representative of the majority of tennis shots for most players. Both the third and fourth videos contained footage of serves. The quality of the third video was much lower than all of the other videos.

<table>
<thead>
<tr>
<th>Video</th>
<th>Time</th>
<th>Number of Bounces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Video 1</td>
<td>2:13</td>
<td>51</td>
</tr>
<tr>
<td>Test Video 2</td>
<td>2:40</td>
<td>22</td>
</tr>
<tr>
<td>Test Video 3</td>
<td>2:45</td>
<td>20</td>
</tr>
<tr>
<td>Test Video 4</td>
<td>3:16</td>
<td>29</td>
</tr>
</tbody>
</table>

5.1.2 Ground Truth

Ground truth needs to be obtained for comparison with the results obtained. It can then be used to evaluate the performance of the solution. The ground truth for these testing videos had to be determined manually.

This had to be done by stepping through the videos frame by frame. The bounce point is then determined manually by clicking the position that the tennis ball bounces in the frame that it bounces. This would output the pixel location and the frame number of the bounce point which can then be used as ground truth for that bounce point. This method of obtaining ground truth involves human input and therefore introduces human error. It is not always clear exactly where the bounce point is as the actual bounce is between frames in a lot of cases. The user therefore needs to determine where the bounce point is based on the flight path. Figure 5.1 below shows two successive frames before and after a bounce point. The shadow of the tennis ball can be seen away from the tennis ball in each frame. It is therefore not touching the ground in either of these two frames. When the tennis ball is touching the ground, the shadow can be seen directly next to the ball, as can be seen in Figure 5.2 below. It is therefore up to the discretion of the user to decide on a bounce point for this particular shot in Figure 5.1 as it cannot be directly seen in the video.
This human input leads to inaccuracies in the ground truth. There are a couple of scenarios where these inaccuracies are accentuated. The first of these is when the shots are moving very quickly. In this case, the ball may not be close to touching the ground in any frame. It is therefore difficult to determine where the tennis ball bounced with a high accuracy. The
second of these scenarios occurs in the third test video. The quality of footage in this video is much lower than the other videos. As a result, it is more difficult to accurately locate the bounce points of the tennis shots in this video.

Testing for the bounce points is done in the video’s image space and not in the transformed space of the shot placement report. Therefore, the ground truth only needs the pixel location of the bounce in the video. The bounce point does not need to be obtained in the court diagram image space. There would be no way to accurately obtain ground truth for the bounce point in this transformed space. The transformation is therefore tested separately. The evaluation of the implementation to detect bounce points can therefore be done in the video’s image space.

5.1.3 Testing Metrics

Testing Metrics need to be calculated in order to evaluate the implementation. These metrics provide a numerical value indicating the performance of the system in each of the test videos. There are four simple metrics when it comes to evaluating a solution. [13]

![Figure 5.3 Simple Testing Metrics Table](image)

The first of these is the True Positive. A true positive in this scenario is when a bounce point is detected correctly. The bounce point found is within a certain tolerance for both time and location when compared to the ground truth. The tolerance for the time of the bounce point is two frames. The tolerance for the location of the bounce point is 40 pixels. Therefore, bounce points detected by the solution need to be within 40 pixels in distance and within two frames in order to be considered a false positive.
When the bounce point is determined to be a true positive a second metric is calculated. The first metric is the count of the amount of true positives. The second metric is the average error of the solution. The distance between the bounce point found and the ground truth is located. The average distance is then calculated for all of the bounce points correctly found in the video. This results in the **Average Error** for that video. This is needed in order to evaluate the accuracy of the solution even when the bounce points are found correctly.

The second of the basic metrics obtained is the **False Positive**. In this application, a false positive constitutes a bounce point that was incorrectly found by the solution. The bounce point found does not match to any of the bounce points in the ground truth for the video. It is not within the tolerance for at least one of the frame or the distance. This metric therefore acts as a measure of how many bounce points the solution finds that are not actually there.

The last of the simple metrics calculated as a part of this testing process is the **False Negative**. In this case, false negatives are bounce points in the ground truth that were not found by the solution. None of the bounce points found in the video were within the tolerances for time and distance when compared to the ground truth. Therefore, the bounce points were not found. The metric measures the amount of bounce points that the solution missed in the video.

The last of the simple metrics is the True Negative. A true negative in this case would mean that the solution did not locate a bounce point in a time and location where a bounce point did not exist. In this application, this scenario occurs a near-infinite amount of times. As a result, the true negatives are not calculated as a part of the evaluation process of this application.

At this point, there are four separate metrics being used in order to evaluate the solution. These include:

- True Positives (TP)
- Average Error (AE)
- False Positives (FP)
- False Negatives (FN)

A further two metrics are also calculated in order to evaluate the solution. The first of these is **Recall**. It is defined by the formula:

\[
\frac{TP}{TP + FN}
\]
The true positives and the false negatives together represent all of the bounce points in the ground truth for the video. This metrics therefore shows the percentage of the actual bounce points found in the video that are detected by the solution. This metric indicates how robust the system is in finding the actual bounce points in the video.

The second metric is **Precision**. This metric is defined by the formula:

\[
\frac{TP}{TP + FP}
\]

The true positives and the false positives together represent all of the bounce points found in the video. This metric shows what percentage of the bounce points found in the video by the solution are actually bounce points. This metric provides an indication of how accurate the system is at distinguishing actual bounce points from the false positives.

These two metrics, Recall and Precision, along with the Average Error of the bounce points that are correctly found are the three metrics used in order to evaluate the performance of the system.

### 5.1.4 Implementation

Once the test videos along with the ground truth had been obtained, and the evaluation metrics had been determined, a system was then developed for automatically testing the solution. This system had to calculate all of the testing metrics for each of the test videos in order to evaluate the performance of the system.

All of the videos are looped through. For each video, the ground truth is retrieved for that video. The ground truth is stored in separate text files for each video. The solution then locates all of the bounce points in that video.

These bounce points are then compared with the ground truth for the bounce points in that video. When comparing the bounce points found and the ground truth, both the distance between the two points and the frame number of each point were checked.

If both of these aspects are within a certain tolerance for one of the bounce points found when compared to the ground truth, it is determined that the bounce point was correctly found for that point in the ground truth. The tolerance for the distance is 40 pixels while it is 2 frames
for the time. A count of the bounce points correctly found is kept to represent the true positives. The distance between the bounce point found and the ground truth is added to the total error. This is then used once all of the bounce points in the video have been checked in order to calculate the average error for the video. Finally, the bounce point from the ground truth is marked as being found.

If either of the distance or the frame number are not within the tolerance for any of the bounce points in the ground truth, it is determined that the bounce point found was a false positive. A count for the amount of false positives is also kept.

Once all of the bounce points found in the video have been checked, the bounce points in the ground truth then checked. Any bounce points in the ground truth that are not marked as being found are determined to be false negatives. The total amount of these false negatives is calculated.

The amount of true positives, false positives and false negatives are then used to calculate the Recall and Precision evaluation metrics. All of these metrics are outputted for each of the test videos.

5.1.5 Results

The following results were obtained when the solution was tested for the four different test videos.

<table>
<thead>
<tr>
<th>Video</th>
<th>Number of Bounces</th>
<th>True Positives</th>
<th>False Positives</th>
<th>False Negatives</th>
<th>Average Error</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>51</td>
<td>26</td>
<td>5</td>
<td>25</td>
<td>8.7</td>
<td>51%</td>
<td>84%</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>15</td>
<td>19</td>
<td>9</td>
<td>13.9</td>
<td>63%</td>
<td>44%</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>0</td>
<td>15</td>
<td>20</td>
<td>N/A</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>7.63</td>
<td>38%</td>
<td>48%</td>
</tr>
</tbody>
</table>

As you can see in the table above, none of the bounce points in the Test Video 3 were detected by the algorithm. This video had very low quality footage as mentioned previously. The images are very blurred in this video. It is not representative of the quality of videos taken with cameras on mobile devices. The tennis ball is rarely detected in any of the frames. As a result, no continuous paths are tracked. The bounce points can therefore not be detected.
The false positives in this video occur as there is a lot of movement visible in the background of this video. A yellow and green train moves past at various stages during the video resulting in motion being detected.

![Figure 5.4 Example frame from Test Video 3](image)

The Recall metric only varied a small amount in each of the other three videos. The tracking of the tennis balls was successful in the majority of cases. When bounce points were not detected, it was due to the ball not being detected in a few frames close to the bounce point. The ball was still tracked in for the majority of the flight path in most cases. Failure to detect the tennis ball in a frame usually occurred when the ball was further away from the camera. Therefore, bounce points that occurred further away from the camera had a lower chance of being detected.

On the other hand, the Precision metric was much greater in Test Video 1 than in Test Video 2 or Test Video 4. This was because there were far more false positives in these two videos. The solution detected bounce points in these videos that were not actually bounce points. There were a couple of different reasons for this.

In Test Video 2, the player was standing closer to the camera than in Test Video 1. As a result, the movements of the player were clearer and detected by the solution much more regularly. Also, the player moved across the court in this video to pick up some tennis balls which resulted in a path being tracked. Furthermore, a few of the shots hit the net in this
video. There were no bounce points obtained as ground truth for these shots, but they still bounced off the ground after hitting the net. These bounce points were occasionally detected depending on whether the tennis ball was occluded by the net at this time. Finally, there was more movement in the camera in this video. The shots in this video were being hit with more power and as a result, the camera would move more when the fence it was attached to was hit.

This same issue with the camera movement was also present in Test Video 4. It was more pronounced in this case as the ball often hit the fence closer to the camera in this video, resulting in even more motion. In this video, the player was further away from the camera, at a similar distance to Test Video 1. However, this video contained footage of the tennis player hitting serves. As a result, there was a lot more motion along the y axis in the video from the player. Due to the fact that bounce points were detected by looking at motion in the y axis, this resulted in some bounce points being detected here that were not actual bounce points. Furthermore, for some of the bounce points that were not detected by the solution, the second bounce of the shot was detected. The solution only accepts the first bounce points on a path. Since the first bounce on the path was not detected, the solution determined that the second bounce was the first bounce point on the path and accepted it as a bounce point.

5.2 Court Location

Part of the solution to this application involved locating the court automatically. This involved locating the lines on the court and then locating the line intersections. This implementation was tested by analysing the results of the solution and comparing them to the ground truth.

5.2.1 Ground Truth

Ground truth for this testing scenario was easier to acquire than the ground truth for the bounce points. The reason for this was because the implementation only worked on one image rather than a video sequence.

Various images of different tennis courts were obtained for the ground truth. For each of these images, the pixel locations of the line intersection points were identified. These acted as the ground truth for the image.
5.2.2 Implementation
A similar testing system that was implemented for testing the detection of bounce points was also used in this scenario.

For each of the images, the solution detected the court’s location. The line intersection points found by the solution were compared to the ground truth. These points were then compared to the ground truth.

Similar metrics that were used for the evaluation of the bounce detection could have been used for this evaluation also. However as previously mentioned, the implementation did not correctly locate the intersection of the court’s lines. Therefore, none of the line intersection points from the ground truth were located. As a result, no metrics were able to be calculated.

5.2.3 Results
Unfortunately, there were no results for this section of the solution due to the aforementioned problems.

6 Conclusion

6.1 Review of Application
The goal of this project was to develop a system for a mobile application that would track a tennis player’s shots and provide them with a report of their shot placement.

The tennis balls were detected in a single frame by first detecting both the moving pixels and the yellow pixels. These were clustered together to form moving objects and were filtered in order to obtain the moving candidate tennis balls in the frame.

Once the candidate tennis balls were detected, they were then matched to the existing ball flight paths using the prediction for each path or the proximity to the last point in the path. If a candidate tennis ball did not match any existing path, a new one was created using that ball as the starting point. If no new balls were matching to a path, that path was then deleted.

The bounce points were detected by first detecting a change in direction in the path. Two points were taken from both the downward trajectory and the upward trajectory in order to
form two lines. The intersection of these two lines was then calculated in order to locate the bounce point.

The bounce points found were then transformed onto a separate image of a court diagram to form the shot placement report for the players. This geometric transformation was calculated by matching the line intersection points on the court found in the video with the corresponding line intersection points on the court diagram.

### 6.2 Conclusions

The system is not working in real-time. The two main components that seem to have the most effect on the efficiency of the solution are the blob detection and the RANSAC method. This is not ideal as an application using this system would need to close to real-time in order to be useful. Despite this, the tracking of the tennis balls could still be useful. While it cannot be used to immediately show a player a report of their shot placement, it can be shown to them later for analysis.

The efficiency of the solution is affected a lot more when there are a lot of blobs found in the image. This occurs when the camera moves, creating a lot of moving pixels and objects. It would therefore be very useful if the video was stabilized before the motion in the frame was detected.

The system tracks the ball flight paths in the majority of cases. The reason for bounce points being missed by the solution is generally that the tennis ball was not found in a few frames close to the bounce point. For that reason, a more robust system for handling missing frames would be useful.

### 6.3 Limitations

Due to the quality of cameras on mobile phones, it is unrealistic to develop an application that can detect bounce points for the entire court at one time. Therefore, for this application, the objective was to only look at one half of the court. It is unrealistic to look at the entire court as if the camera is placed far enough away to have the entire court in view, the tennis balls would be too small to track accurately.
The original goal of this project was to design a system for a mobile application to determine if the tennis ball had bounced inside or outside the court similar to the Hawk-Eye system used. However, it was determined early on in the project that this goal was unrealistic with a mobile phone camera due to the accuracy needed in detecting bounce points for this type of application.

6.4 Further Work

There are a couple of different aspects of the solution that could be improved on and worked on in the future.

The first aspect of the solution that should be improved is its efficiency. It is not currently working in real-time and would therefore need to be optimised. This application is not very useful if it is not working in real-time or at least close to real-time.

The next aspect to improve would be to stabilize the camera. As mentioned, the solution does not perform as well when the camera moves. The motion in the camera affects the solution’s ability to detect motion in the image accurately. It results in a lot more moving objects being detected and in some cases, bounce points that are not the bounce points of actual tennis shots are detected. By stabilizing the camera, the movement in the camera would be alleviated, and the moving pixels in the image could therefore be detected more accurately.

As mentioned previously, the court was not detected automatically. This is another feature that could be implemented in the future. Another feature that could be implemented is adding the tracking of players to the solution. This could be used to help determine if the ball has been hit. It can also be used to determine whether the players have switched sides when there are two players playing so you know which player is hitting to which side of the court. If you want to provide a player with a shot placement report, and they are switching sides, you need to know which shots are theirs.

One feature that can vary in different cases is the colour of the ball. The colour of the ball in the video does not only depend on the actual colour but also on the lighting conditions. Another feature that could be implemented in the future is that the solution would automatically determine the colour of the ball. It could learn the colour of the ball by determining which of the moving objects were tennis balls and then examining their colour.


[7] OpenCV, SimpleBlobDetector, April 2021. URL: https://docs.opencv.org/3.4/d0/d7a/classcv_1_1SimpleBlobDetector.html


