Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Luke Gibson

9th May 2020

Ball Tracking to Determine Out-Line Decisions in Squash

Project Supervisor: Dr. Sasan Mahmoodi

Second Examiner: Prof. Tim Norman

A project report submitted for the award of
Bachelor of Computer Science

Abstract

Technological advancements have allowed many sporting organisations to supplement referee decision making processes with the use of computer vision systems, resulting in fairer and more accurate outcomes for players. The sport of squash poses unique problems for these systems due to the small size and high speeds of the ball, making it hard to track.
This report documents the development of a computer vision ball tracking system aimed at determining out-line decisions in the game of squash from a single camera view. The system combines the processes of background subtraction and shape descriptor filtering, using consistent object features ensured by environmental constraints to detect the ball and produce a 2D track. An evaluation of the system's performance shows computer vision can be used in squash to produce out-line decisions with the accuracy of a qualified referee.

Statement of Originality

I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.

I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.

I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

Acknowledgements

I would like to thank my supervisor, Dr. Sasan Mahmoodi, for his advice and support throughout this project's development. Furthermore I would like to thank Professor Tim Norman for his valuable input. Finally I would like to express my thanks to my friends in the Southampton University Squash Club for their participation in my project data collection.

Contents

1 Project Statement **8**
- 1.1 Problem Description . . . 8
- 1.2 Project Goals . . . 8
- 1.3 Project Scope . . . 9

2 Literature Review **11**
- 2.1 Existing Systems . . . 11
  - 2.1.1 Hawk-Eye Tennis System . . . 12
- 2.2 Image Processing Techniques . . . 13
  - 2.2.1 Background Subtraction . . . 13
  - 2.2.2 Object Detection . . . 14
  - 2.2.3 Object Tracking . . . 15
- 2.3 Squash Information . . . 16
  - 2.3.1 Ball Behaviour . . . 16
  - 2.3.2 Detection constraints . . . 17

3 Design **18**
- 3.1 System Requirements . . . 18
- 3.2 System Description . . . 18
- 3.3 Interface . . . 20
- 3.4 Design Decisions . . . 21

4 Implementation **22**
- 4.1 Object Detection . . . 22
  - 4.1.1 Out-Line Detection . . . 22
  - 4.1.2 Ball Detection . . . 23
  - 4.1.3 Ball Prediction . . . 24
- 4.2 Contact Detection . . . 24
  - 4.2.1 Trajectory Features . . . 25
  - 4.2.2 Final Method . . . 26
- 4.3 Calculations . . . 26
  - 4.3.1 Impact Radius . . . 26
  - 4.3.2 Ball Out Probability . . . 27
- 4.4 Interface . . . 30

5 Testing **31**
5.1 Data Set . . . 31
5.1.1 Data Variation . . . 31
5.2 Contact Detection . . . 32
5.3 Decision Accuracy . . . 33

6 Evaluation **36**
6.1 System Evaluation . . . 36
6.1.1 Object Detection . . . 36
6.1.2 Contact Detection . . . 37
6.1.3 Out Probability Model . . . 37
6.1.4 Performance . . . 38
6.2 Comparative Evaluation . . . 39
6.2.1 Process Comparison . . . 39
6.2.2 Accuracy Comparison . . . 40
6.2.3 Cost Comparison . . . 40
6.3 Project Management . . . 41

7 Conclusion **42**
7.1 Project Goals . . . 42
7.2 Future Work . . . 42

8 References **44**

Appendices **48**

A Appendix A **48**
A.1 Initial Gantt Chart . . . 48
A.2 Final Gantt Chart . . . 48
A.3 GitHub Commits . . . 49

B Appendix B **50**
B.1 Application Interface: Initial . . . 50
B.2 Application Interface: Processing . . . 51
B.3 Application Interface: Decision . . . 52

C Appendix C **53**
C.1 Participant Information Form . . . 53
C.2 Participant Consent Form . . . 54

D Appendix D **55**
D.1 Project Archive Contents . . . 55
D.2 Original Project Brief . . . 56

List of Figures

1.1 International squash singles court dimensions [3] 9

2.1 Hawk-Eye ball tracking pipeline for tennis [7] 12
2.2 Effect of noise reduction in background subtraction [10] 14
2.3 Categories of object tracking [10] 16

3.1 System processing pipeline 20
3.2 User interface wireframe 21

4.1 Effect of contour filtering 23
4.2 Undetected ball over out-line 24
4.3 Ball prediction model 24
4.4 Angle between vectors 25
4.5 Gradient change oscillation after contact with wall 26
4.6 Graph showing how impact radius percentage scales with compression 27
4.7 Effect of morphological closing on ball imprint area 28
4.8 Algorithm to convert out-line contour to out area 28
4.9 Detection probability represented as brightness 29
4.10 How probability value relates to shot in/out decision 30

5.1 A sample of camera orientations used in the shot dataset 31
5.2 Contact detection accuracy of system 33
5.3 Out probability accuracy of system 35

6.1 Example of shot prediction during contact resulting in incorrect decision 37

List of Tables

2.1 Ball tracking challenges across different sports [5] 11
2.2 Examples of commercial ball tracking systems [4] 13

3.1 System requirements 18

4.1 Possible pixel values in imprint and out area weighted sum . . . 29

5.1 Contact detection test results 33
5.2 Out probability test results 34

6.1 Evaluation of system requirements 38

Project Statement

1.1 Problem Description

Squash is a faced paced racket sport in which "over 20 million players participate regularly world-wide in over 185 countries" [1]. Figure 1.1 shows the squash court out-line, labelled as 'wall line', which marks to court boundary on each wall. During a rally, if a player's shot results in the ball making contact on, or above, the out-line the player loses the rally, it is the duty of the referee to call the shot as 'out' in this case.

Currently the Professional Squash Association (PSA) rely on a single referee, located 5-10m behind the court, to detect these shots. Due to the balls high speed and small size, measuring only 40mm in diameter, the out-line decisions made by the referee are often disputed by players and spectators [2][3]. Incorrect decisions may end up dictating the result of a match and the prize money taken by the players. Over the last couple of decades multiple sporting associations have integrated computer vision tools into the refereeing process, for example the tennis Hawk-Eye system and football Goal-Line Technology [4]. For squash to remain credible in the world of professional sport it is important it too adopts technologies to reduce inaccuracies in the refereeing process, a sentiment expressed by professional players seen in the below responses received when asked their opinion on computer aided decision making in squash.

"It's generally because it's so hard to see on the replay ... players would obviously like more conclusive options." - Sarah-Jane Perry, world no.7

"That sounds just what we need. Purely because SquashTV (broadcast system) *can't cater for it. Its never clear so they can't rely on it."* - Joey Barrington, PSA world tour commentator.

1.2 Project Goals

This project aims to produce a computer vision system to aid the referee's out-line decision making process. Referee's could then use this as a tool to review disputed out-line decisions, supplying a video clip of the disputed shot and receiving a more reliable decision from the system. The final system will aim

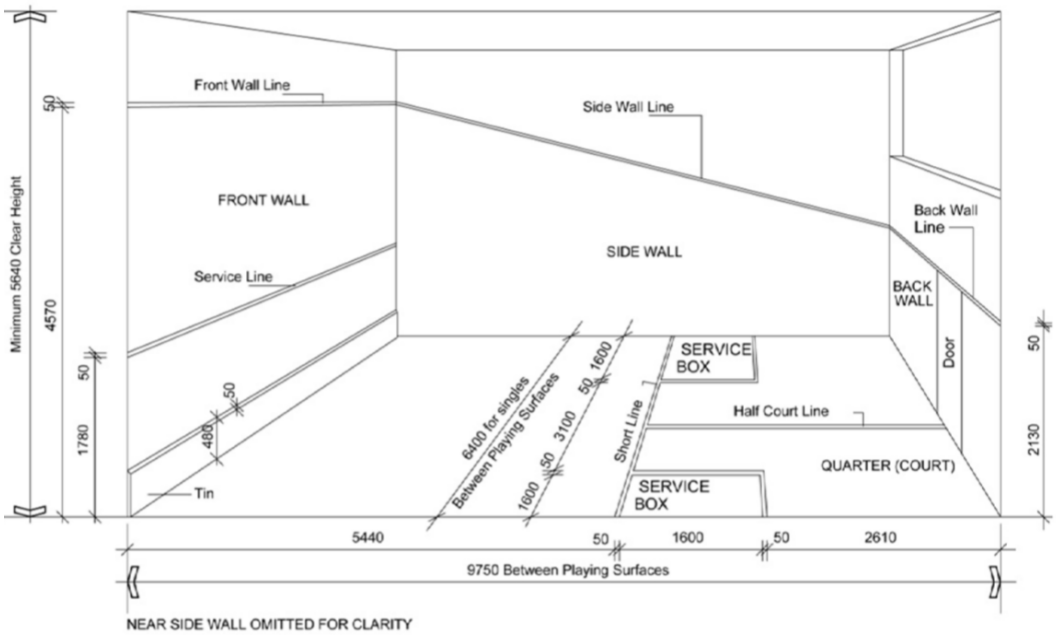


Figure 1.1: International squash singles court dimensions [3]

be simple and intuitive as the intended user will be a non-technically trained squash referee.

This project will investigate how different image processing operations can be applied to the problems of ball and line detection, ball tracking and contact detection. When implemented these processes will be measured to ensure the accuracy is greater than that of a single unaided referee.

1.3 Project Scope

This project will be limited by the following factors:

1. **Budget** - Due to a small project budget the system will be restricted to a single camera feed when processing a shot as having multiple camera's would require the syncing of camera frame rate's which is a feature not found on camera's within the budget.

2. **Data** - As no large dataset of high speed squash footage exists the project will require me to collect footage myself. This will reduce the feasible dataset size I will be able to use when testing the system.

3. **Access** - As I will not have access to an official PSA world tour court the project will be designed around the university squash court with the assumption the system could be translated to other courts in the future.

4. **Time** - Due to the limited time I have throughout the 2019/20 academic year, alongside my other modules, I will only be looking at calculating

out-line decisions on the court's side walls. Collecting shot data at the front and back wall out-lines would require an additional time investment without providing a different challenge for the system.

5. **Processing Power** - Due to the limited budget the system will be developed with the constraint of running in reasonable time on my personal laptop. This will limit the resolution of the collected data as to keep processing times reasonable.

Literature Review

In this section I investigate current computer vision systems used in sport as well a more detailed review of the different object detection and tracking operations that make up such systems. This literature will help in deciding how I design my system as to overcome the challenges its faces.

2.1 Existing Systems

Current computer vision systems are vital to the way modern day professional sport operates. The data collected via these systems is used during play in the form of referee assistance and broadcast statistics as well as behind the scenes for training and coaching [4][5]. To collect this data accurately tracking ball movement from video sequences is vital [5]. Exactly how a system achieves this is largely dependent on the sport it is designed for, as each sport posses a unique set of challenges. The main challenges systems may face include "occlusion, misdetection, illumination variation, different quality of videos with varying frame rates and real-time constraint" [5]. How these apply to different sports is highlighted in Table 2.1.

Sport	Dominant Features	Challenges
Pool, Billiards, Table Tennis	Small region needed, ball colour, less cameras needed	Small ball size, varying ball velocity, non-linear non-periodic oscillatory motion
Tennis	Court Lines	Medium sized ball
Golf	Little or no occlusion, less camera's needed	High ball velocity
Volleyball, Basketball, Football	Bigger ball size, slow ball velocity, lines on ground	same and opposite team occlusion, players can possess the ball for a long time
Hockey, Cricket, Baseball	Lines on the ground	Medium ball size, occlusion, high ball velocity
Rugby, American Football	Lines on the ground	Very heavy occlusion, irregular trajectories, spheroidal ball shape

Table 2.1: Ball tracking challenges across different sports [5]

The video sequences used by these systems will contain noise that is not required for the task of ball tracking. A large part of this noise will be in area of the video sequence that contains non-play objects such as the surrounding stadium and spectators. Many systems opt to first remove this area, effectively extracting the useful area of play before removing remaining noise in the video sequence [5].

The number of different video sequences a system uses will determine what methods can be used for ball detection. The main advantage of a system that uses multiple cameras in different locations is the ability to use "3D localisation of the ball and players" [5]. This heavily reduces the problem of occlusion but usually requires the systems camera to be calibrated and fixed in a particular location [4].

2.1.1 Hawk-Eye Tennis System

Tennis's Hawk-Eye ball tracking system was "one of the first commercially available multi-camera systems" employing "6 to 10 high performance cameras" to track the ball in 3D space [4][6]. The system pipeline is described below [4][7].

1. Uses an expected size to filter possible ball candidates in the image from each camera.
2. 2D tracklets of the ball's motion are built up on the image plane.
3. A 3D reconstruction module the reconstructs these into a 3D tracks.
4. Impact point is determined between these tracks.
5. The centre of lines are then used to determine the line position as the observed width of lines varies due to image effects.

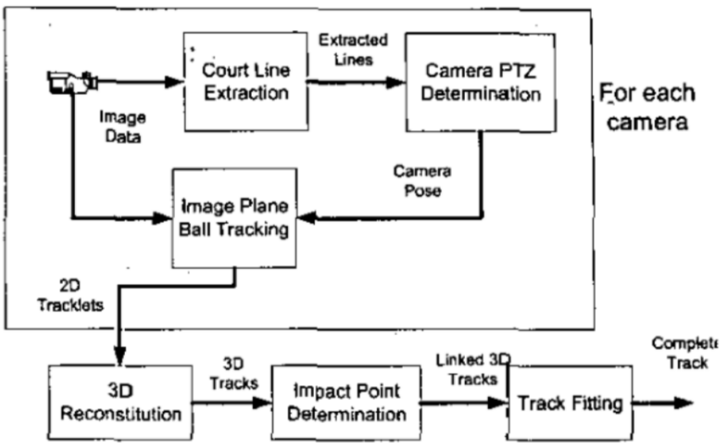


Figure 2.1: Hawk-Eye ball tracking pipeline for tennis [7]

The systems ball detection is stated as having a "mean error of only 2.6mm" when measured against a ground truth found from placing a high speed camera directly on the court [8]. The system has been appreciated by both tennis spectators and broadcasters as well as reducing the pressure on officials and players due to the ability to check decisions via a replay review [4][9]. However due to hardware requirements of the system it is only accessible in tournaments with a large budget, so can not be adopted in all professional tennis matches [6].

Since the introduction of this system, similar systems have been implemented in other sports as their governing bodies are pressured to maintain the highest levels of decision accuracy [4]. Table 2.2 shows a summary of some of these systems.

System	Camera Configuration	Precision & Performance
Hawk-Eye Cricket	Six 340fps cameras	5-10mm accuracy. Ball size in image is circa 10 pixels and centre is found to circa 1/3 pixel
Hawk-Eye Tennis	Up to ten 340fps cameras	Mean error of 2.6mm when compared to a high speed camera located on the playing surface
Hawk-Eye Goal-Line	Seven cameras per goal	FIFA requirements include a positional accuracy of +/- 1.5cm and that indication of a goal is proved in 1 second or less

Table 2.2: Examples of commercial ball tracking systems [4]

2.2 Image Processing Techniques

2.2.1 Background Subtraction

This technique involves finding the absolute difference between the current image $I(x, y)$ and a background image $B(x, y)$, producing a resultant image $F(x, y)$ containing only the foreground objects of the original image [10]. The foreground objects are detected as a significant difference from the background "indicating these as the pixel locations of moving objects" [11]. A binary image is produced using

$$F(x, y) = \frac{|I(x, y) - B_t(x, y) - \mu|}{\sigma} > T$$

where foreground pixels are represented by a 1, and background pixels are represented by a 0 [12]. To work as intended the background image must contain no moving objects and be similar to the scenes current brightness and position [11]. For this reason a dynamic background model is often used, created via the difference found in the frames neighbouring the current frame. In a video

sequence you can initialise this to use "the first frame as the background directly or, the average pixel brightness of the first few frames" [11].

Background subtraction is susceptible to extracting significant levels of noise in the foreground image. This can be due to factors such as camera motion, varying lighting conditions, motion blur and unwanted detection of moving shadow regions [12][11]. Figure 2.2 shows how this can be addressed in a binary image with "expansion and corrosion operations", eliminating small disconnected areas of noise [11].

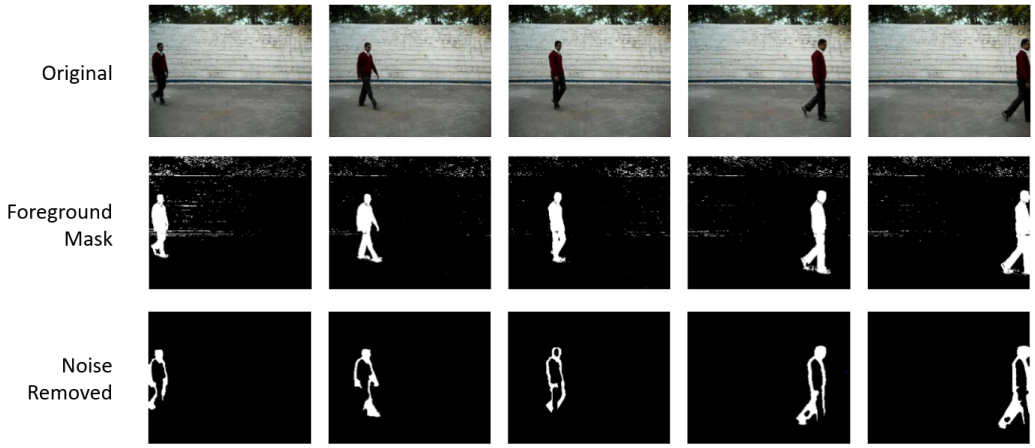


Figure 2.2: Effect of noise reduction in background subtraction [10]

2.2.2 Object Detection

Detecting particular objects in an image can be achieved in many ways, generally the methods aim "at discrimination of the target from the background" [13]. This is achieved using shape descriptors to filter the object candidates of the foreground blobs. The descriptor used is dependent on the object the system intends to detect, possible descriptors include the objects: perimeter P, area A and roundness $P^2/4\pi A$ [13].

Thresholding an image based on the pixel values is another way to detect objects, and is useful if the object being detected is of a unique colour to the other candidates in the image foreground. Grayscale image thresholding defines a range of brightness such that only the pixels of the original image within the range are accepted, and all other pixels are rejected [14]. The formula

$$X(x,y) = \begin{cases} 1, & \text{if } T_1 \geq I(x,y) \leq T_2 \\ 0, & \text{otherwise} \end{cases}$$

defines the resultant binary mask of grayscale thresholding X where T_1, T_2 denote the lower and upper bound of brightness respectively and $I(x,y)$ denotes the brightness of the pixel in the original image.

Colour image thresholding works in a similar way, but due to more than one variable characterising each pixel the threshold must provide a range for each colour channel of the original image [14]. For a pixel of the original image to be accepted each of its channel values must be within the range specified for that channel. This is "logically equivalent to segmenting each plane individually, creating separate binary images and then combining them with a Boolean AND operator afterwards" [14]. The above formula can be then be adapted to account for an arbitrary number of channels where C denotes the set of channels in the original image.

$$X(x,y) = \begin{cases} 1, & \text{if } \forall c \in C : T_{c1} \geq I_c(x,y) \leq T_{c2} \\ 0, & \text{otherwise} \end{cases}$$

Detectors can be combined to create an enhanced single detector [13]. For example applying shape descriptor filters to the remaining candidate objects of a previous filter creates a stricter, more precise detector, where each additional step enhances the detection before.

2.2.3 Object Tracking

Once an object is detected within an image an object tracker uses its movement between successive frames of a video sequence to generate its trajectory, "traditionally utilising the assumption of temporal continuity" [10][13]. Different preprocessing steps may be used depending on the systems application, these include [10]:

- Convert to grayscale - Reduces processing time as operations are only performed over a single colour channel instead of RGB's three colour channels.
- Smoothing - Applying a median filter reduces the noise in the original image, increasing the accuracy of moving object detection.
- Reduce resolution - As well as reducing processing time due to few image pixels this will reduce the likelihood of detecting 'fake motion' such as swaying tree branches.

Figure 2.3 shows the different methods of object tracking which fall into three main categories. The method used is dependent on the application of the system it is being designed for.

Point tracking - Object track is generated from the (x, y) points that represent it across consecutive frames [10].

Kernel tracking - Object track is generated from the "parametric motion or the dense flow field" of the simple object region that defines it across consecutive frames [10].

Silhouette tracking - Object track is generated by finding the object region in the current frame by matching a model region generated using the previous frames [10].

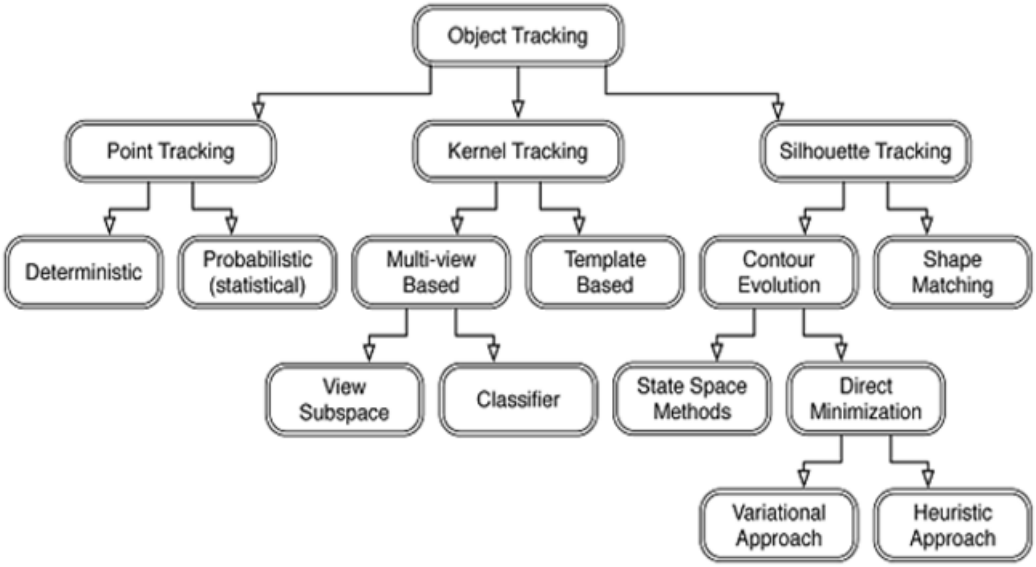


Figure 2.3: Categories of object tracking [10]

In the case a frame of the video sequence has no detection of the object being tracked a model must be used to predict the location of the missing detection. Detentions are often missed due to object occlusion, when the line of sight between the camera lens and the desired object is broken.

Kalman filtering is an approach to this prediction model which estimates the velocity and acceleration of an object from its previous locations throughout the video sequence [5]. However, this model assumes linear motion, resulting in low accuracy predictions if the tracked objects motion is non-linear [5].

2.3 Squash Information

2.3.1 Ball Behaviour

The squash ball's behaviour is unique due to its very low rebound resistance, "its ability to absorb energy on loading and release it on unloading" [15]. During play when the ball makes contact with the wall most of its energy is stored elastically in the rubber before rebounding off the wall converting this into kinetic energy. The amount of energy the squash ball has lost, thermally or as sound, during an impact is defined by its rebound resistance which is related to its coefficient of restitution (COR) [15]. COR of a ball is defined by the formula

$$COR = \sqrt{\frac{h_1}{h_2}}$$

where h_1 and h_2 are the drop height and rebound height of the ball [15].

2.3.2 Detection constraints

Particular detection constraints have been shown to work for detecting player and ball objects from broadcast footage of squash in a system designed by Sachdeva [16]. Initially, using pre-configured boundaries, the region of play can be used discarding objects outside this region. From here size constraints can be used to detect the players, as they are the largest contours in the image foreground. Due to the small size of the ball its contour size is similar to that of background noise, meaning size can not be used as a discriminating factor to detect the ball. Instead a velocity constraint can be used as the ball object will be moving a large distance between frames while noise objects will not, allowing the true ball object to be distinguished [16].

Design

This section outlines the planned system implementation along with decisions that have been made as to the tools that will be used to achieve this.

3.1 System Requirements

Table 3.1 shows the systems functional and non-functional requirements denoted in the ID column with 'F' and 'NF' respectively.

ID	Requirement	Definition
F1	Single Camera	Use a single camera angle of a shot to calculate its in/out decision
F2	Object Detection	Detect both ball and out-line objects within an image of 1280 x 720 resolution
F3	Contact Detection	Isolate the moment of a shot when the ball makes contact with the side wall
F4	Decision Accuracy	Calculate in/out decisions with an accuracy greater than that of a single qualified referee
F5	Decision Feedback	Feedback the calculated shot in/out decision to the user
NF1	Performance	Process a supplied shot video within 30 seconds on a consumer grade laptop
NF2	Usability	Intuitive interface allowing easy interaction for non-technical users

Table 3.1: System requirements

3.2 System Description

A dataset containing video clips of squash shots will be collected at Southampton university squash courts via recordings made with an OPPO Reno smartphone. This dataset will be used during the development process to test the image processing pipeline. Figure 3.1 shows the following processing stages performed on a shot from the dataset once loaded into the system.

Ball Detection

For each individual frame of a video sequence the ball object will be found, foreground extraction will be used to achieve this as the ball object will always be moving between frames. To determine the ball from other objects that may also be in the foreground, such as players, shape and motion descriptors will be used to decide which of the foreground object is the most likely ball candidate.

Out-Line Detection

The out-line object will also be found in each frame of a video sequence, colour thresholding will be used to achieve this as the out-line is a solid red colour which highly contrasts the surrounding white wall colour. To determine the out-line object from other objects that may be of a similar colour, such as other court lines and player shirts, shape descriptors will be used to decide the most likely candidate.

Contact Detection

Once the ball has been found in each frame of the video sequence the frames in which the ball makes contact with the wall will be found by mapping the ball trajectory from the detected ball objects. The ball trajectories across the dataset will be analysed to find reliable features, derived from ball location data, that indicate the balls contact with the wall. To estimate ball location when no ball object is detected is a prediction model that assumes linear motion will be used.

Decision Calculation

Once the contact frames of the shot have been found the pixel co-ordinates of the ball and out-line for that frame will be compared. If any of the ball pixels are equal to or above any of the line pixels at the same point along the wall then the shot will be classified as out, otherwise in. This is described by the formula

$$decision(A,B) = \begin{cases} out, & \text{if } \forall(x_1,y_1) \in A, \forall(x_2,y_2) \in B : \exists(x_1 = x_2 \wedge y_1 \geq y_2) \\ in, & \text{otherwise} \end{cases}$$

where A and B are the set of all pixel co-ordinates of the detected ball and out-line respectively at the point of contact. Confidence in the outcome of the decision will be determined by the distance the ball was either over or under the out-line, such that a ball further over the out-line will be decided as out with a greater confidence score.

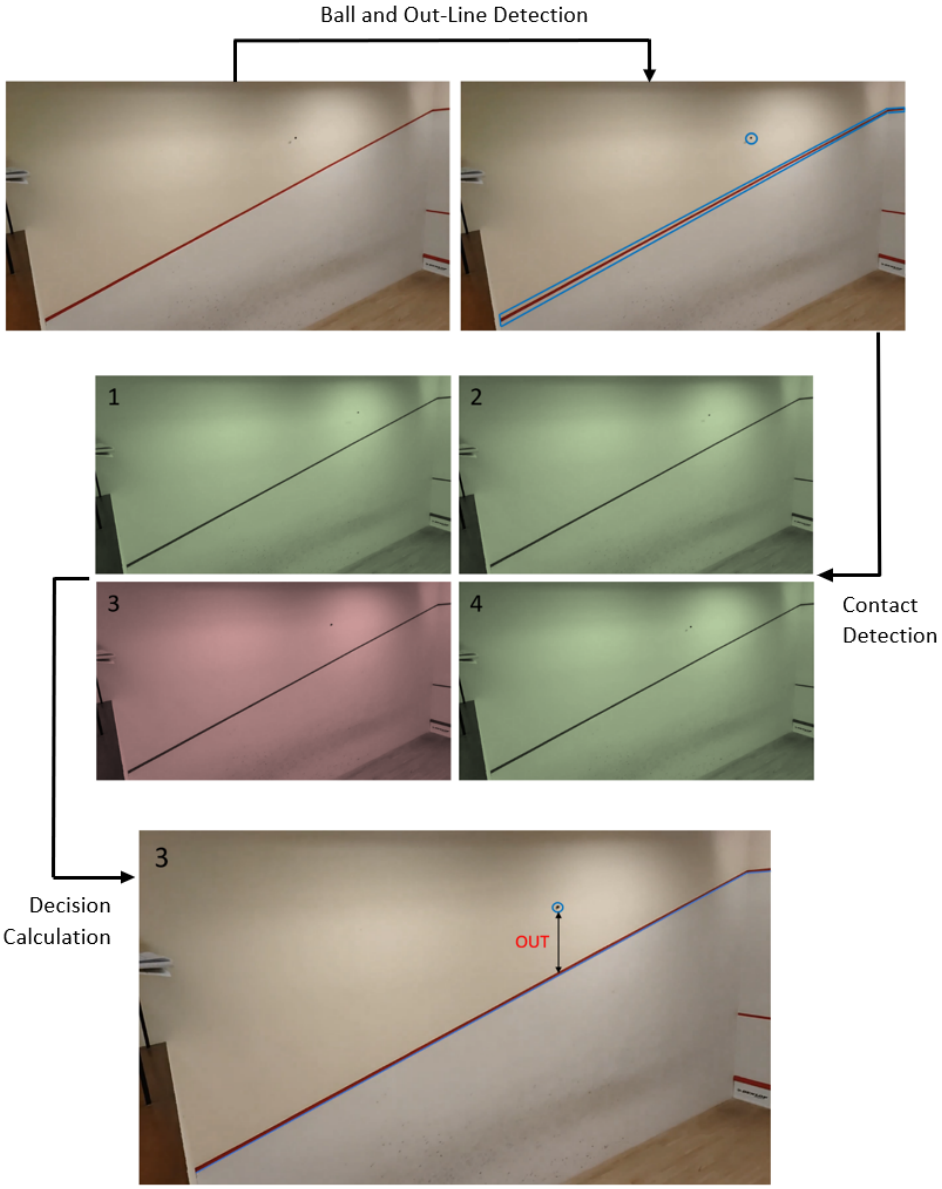


Figure 3.1: System processing pipeline

3.3 Interface

As the system is intended to be used by a referee during a match, to aid operation the processing pipeline will be wrapped in a simple graphical user interface shown in Figure 3.2. The 'Select Shot' button will open a file explorer allowing the referee to select the video clip of the disputed shot, with the 'Make Decision' button starting the video processing and returning the computed decision. A playback screen and controls will allow the referee to see the mapped ball trajectory and detected point of contact. This will allow a referee to spot an obvious detection error is one was to occur.

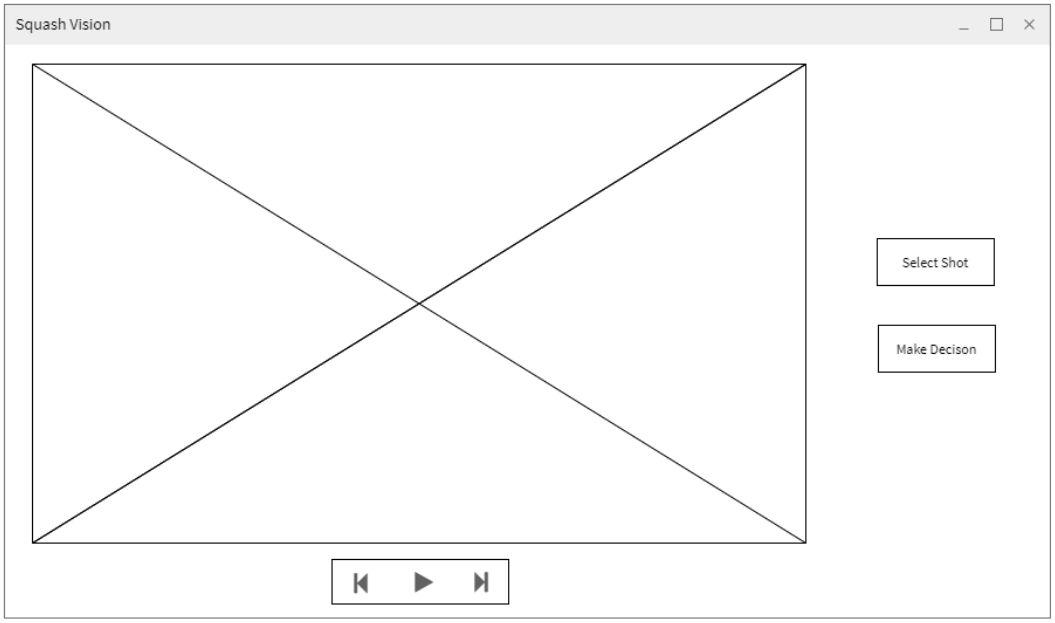


Figure 3.2: User interface wireframe

3.4 Design Decisions

I will implement the system in Python 3 due to my familiarity with the language, its large selection of libraries and their detailed documentation. As I have limited previous experience developing computer vision systems I will be making use of the OpenCV library and its well optimised implementations of image processing algorithms such as foreground extraction and colour thresholding [17]. Using OpenCV's existing implementations will allow me more time to focus on the problems specific to my system.

Python's bundled Tkinter library will be used to create the simple graphical user interface [18]. This has been chosen over other more complex libraries as the requirements of the systems interface are relatively simple. My previous experience with the Tkinter library will result in a quick interface development, allowing me to focus on the image processing aspect of the system.

Implementation

This chapter describes the computational process made by the computer vision system in terms of image operations and analysis on the video sequence of a squash shot. This section will also highlight problems encountered along the way and changes made from the original design outlined in chapter 4.

4.1 Object Detection

Before any calculations are done the object's of interest, ball and out-line, must be individually isolated within each frame of the shot. To achieve this a video MP4 file of a squash shot is loaded into the system using OpenCV `VideoCapture` class allowing the individual frames of the video sequence to be looped through performing the out-line and ball detection processes on each.

4.1.1 Out-Line Detection

The out-line of a squash court must be a different colour to that of the surrounding wall, meaning colour thresholding can be used to isolate this segment of an image. I utilised OpenCV `inRange()` operation and supplied two RGB values to threshold between, in the case of most courts this will be a lighter and darker red colour. The resultant image is then morphological opened using OpenCV `morphologyEx()` to remove noise generated by camera instability.

To be robust against incorrectly detecting other similarly coloured objects, such as player shirts and other court lines, the out-line is then selected as the contour from the thresholding with the largest horizontal span. This is calculated by finding the difference between the minimum and maximum x-values of the contours pixel co-ordinates. This additional filtering stage allows the colour range to be larger which means the system will perform better as environment variables such as brightness vary. The selected contour for the frame is then stored as a list of points as show in Figure 4.1.

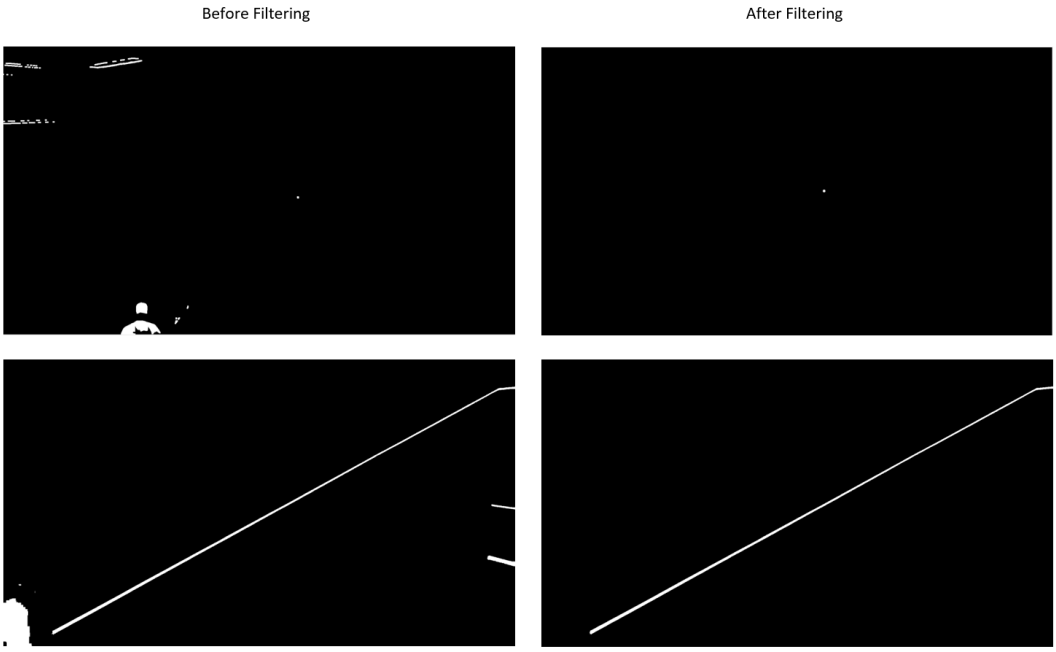


Figure 4.1: Effect of contour filtering

4.1.2 Ball Detection

As the ball is one of few moving objects in the scene it can be detected using background subtraction [19]. To implement this I utilised the OpenCV class `BackgroundSubtractorMOG` which uses a Gaussian mixture-based segmentation algorithm to segment the foreground and background of an image. Similarly to the out-line detection, OpenCV `morphologyEx()` is used to reduce noise in the resulting image.

To avoid detecting players, instead of the desired ball, the individual contours of the image are filtered using detection constraints of a likely squash ball object. Contours with an enclosing circular radius too small or large to possibly be a ball are immediately discarded as possible ball candidates. Remaining contours are sorted using the euclidean distance from the previous frames ball centre to the contours centre in unit of pixels, allowing the contour that is closest to the expected location of the ball to be selected as shown in Figure 4.1. This frames ball detection can then be stored as the centre and radius of the minimum enclosing circle around the contour, preserving the fact that the ball must be spherical.

To avoid incorrectly selecting a non-ball object a minimum threshold on the distance from the last known ball is required, with a no detection flag being stored for this frame to indicate the scenario. Doing this stops the case in which another object would be selected as the ball due to the true ball object not being detected during background subtraction. This was observed at the point in which the ball is in line of sign to the out-line, blending into the background due to their similar colour as shown in Figure 4.2.

Figure 4.2: Undetected ball over out-line

4.1.3 Ball Prediction

Once ball detection has been performed on each frame of the video sequence the frames with no ball detection can have their ball location predicted. It was found that as the ball begins to pass over the out-line the system would still detect the half of the ball not yet 'hidden' resulting in an incorrect detection of the ball. To overcome this the frames neighbouring those with no detection flags had their erroneous half ball detection's overwritten with no detection flags. The detection gaps in the sequence are then found using the no detection flags.

To predict the ball locations within a detection gap the known positions neighbouring a gap are used to find a linear connecting line. Using the length of the gap sequence the missing ball positions are calculated to be evenly spaced between the two neighbouring detection's shown in Figure 4.3. Due to the gaps generally not exceeding 4 frames the assumption of a linear motion is used as the effect of a parabolic trajectory over these distances is negligible.

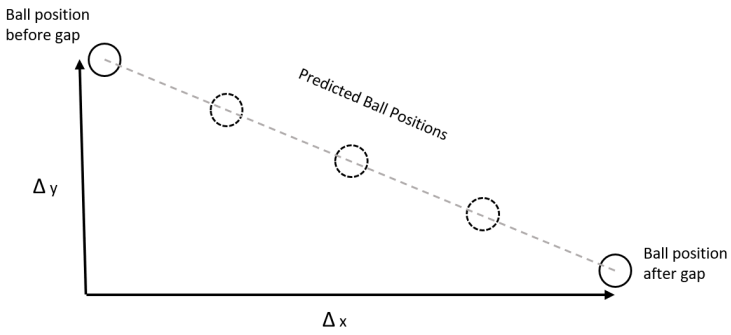


Figure 4.3: Ball prediction model

4.2 Contact Detection

The system predicts the frames of the video where the ball is in contact with the side wall by detecting a patterns seen in the ball trajectory. To find this

pattern I extracted different features and plotted them against frame number to visualise which could be used to detect contact.

4.2.1 Trajectory Features

Ball Speed - A proportional value for this is calculated as the euclidean distance between the current frames ball centre and the previous. Contact is then detected as the frame with the lowest speed this being due to the ball compressing and rebounding from the wall. This feature proved inconsistent at detecting contact in shots where the ball only skims the wall.

Angle Between Vectors - Using neighbouring ball centre's to calculate the two vectors a, b of ball travel before and after the current frame. Figure 4.4 shows how these vectors can be used to calculate the angle at the current frame's ball centre using the formula [20]

$$\alpha = cos^{-1} \frac{\bar{a}.\bar{b}}{|\bar{a}|.|\bar{b}|}$$

This feature was improved by using its deferential to detect contact as the frame with the greatest change in angle value from the previous. However this feature also did not perform well for shots that only skim the wall due to noise from ball detection hiding the angle change due to actual wall contact.

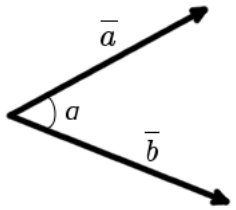


Figure 4.4: Angle between vectors

Trajectory Gradient - The gradient of the tangent to the ball's trajectory at a given point can be calculated from the linear line between the current frames ball centre and a previous frames ball centre. In practice the point 3 frames prior to the current frame was used as it reduced the effect of noise when calculating the gradient using the formula

$$m = \frac{y_1 - y_2}{x_1 - x_2}$$

Again this feature was improved by using its deferential, as well as reducing noise it also makes the feature invariant to camera rotation. Furthermore the gradient values sign would depend on which of the two left and right side walls the shot took place on, using the rate of change normalises this as the change is absolute.

This feature consistently displayed a detectable spike at the frame of contact. However it was observed that often larger spikes would appear after the contact as shown in Figure 4.5. As these spikes only appearing after contact with the wall the system detects the contact frame as the first frame where the features value exceeded a threshold of 0.08. This threshold was balanced such that it was small enough to still detect the slight gradient changes in skim shots while being large enough to not be exceeded in the case the ball did not make contact with the wall.

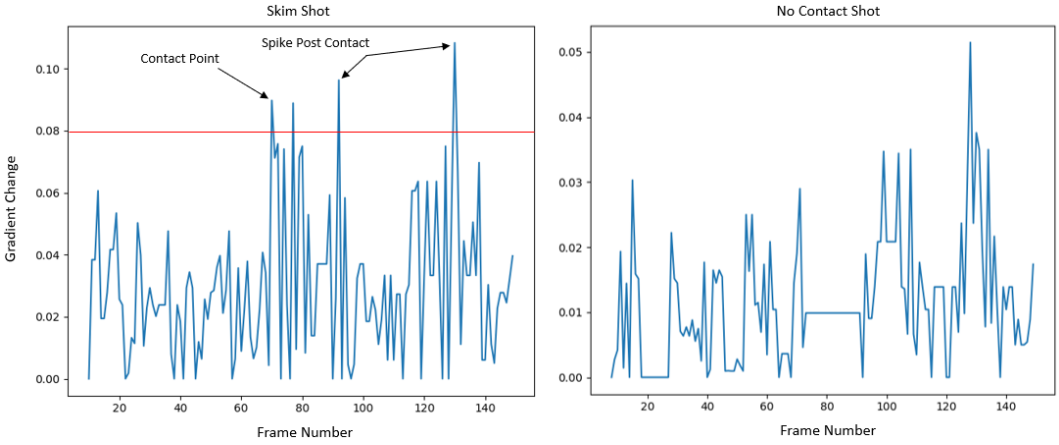


Figure 4.5: Gradient change oscillation after contact with wall

4.2.2 Final Method

Overall the change in the tangent of trajectory gradient was the most consistent way to detect the contact point of the video sequence. For all features averaging each value with its two neighbours reduced the effect of noise present from the ball detection phase. Reducing this noise allowed more constraining thresholds to be used which resulted in the detection of more discrete patterns.

In practice the number of frames in which the ball is in contact with the wall ranged from 2-4 and is linked to the steepness of ball impact normal to the wall such that direct impacts would be in contact for 2 frames while skimming impacts would be in contact for 4. Therefore I used thresholds on the magnitude of the gradient change at the point of contact to be used to determine this.

4.3 Calculations

4.3.1 Impact Radius

During the ball's contact with the wall it compresses such that the imprint radius is some fraction of the ball's full radius. To calculate a value proportional

to this the system will estimate the loss in kinetic energy over the duration of the collision using the formula

$$compression \propto \frac{(s_1^2 - s_2^2)(1+G)}{|F|}$$

where s_1 and s_2 are the ball speed's before and after the collision, G is the gradient change at the frame of contact and F is the set of contact frames [21]. This is adapted from the equation for kinetic energy

$$KE = \frac{1}{2}mv^2$$

incorporating the direction component of velocity change in multiplying the speed change by the magnitude of the gradient change in ball trajectory during the collision. As the mass of the squash ball is constant it can be ignored from the kinetic energy equation.

This value is scaled to represent the percentage of the balls full radius making contact with the wall as shown in Figure 4.6. Thresholds are applied to restrict the minimum and maximum contact percentage, as a squash ball will never compress to completely flat under the power of a shot as observed in high speed footage [22]. This radius percentage changes across the set of contact frames such that the middle frame of contact has the maximum value.

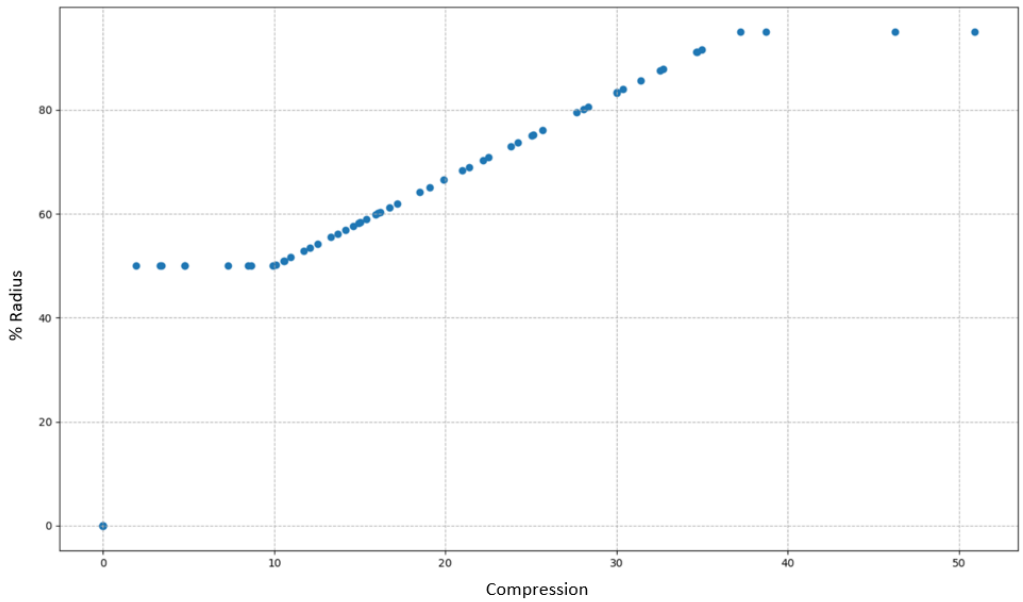


Figure 4.6: Graph showing how impact radius percentage scales with compression

4.3.2 Ball Out Probability

The system calculates the probability a shot was out by summing gray scale images of the out area and imprint of contact area. This operation is only

performed during frames of contact, improving performance, as the imprint will be blank unless contact has occurred.

The imprint area image is generated by drawing the contact area of the ball in each contact frame on a blank image, using OpenCV `circle()` operation supplying the detected ball's centre and radius multiplied by the impact percentage previously calculated. This is performed for each contact frame before being morphologically closed to create a continuous imprint area as show in Figure 4.7.

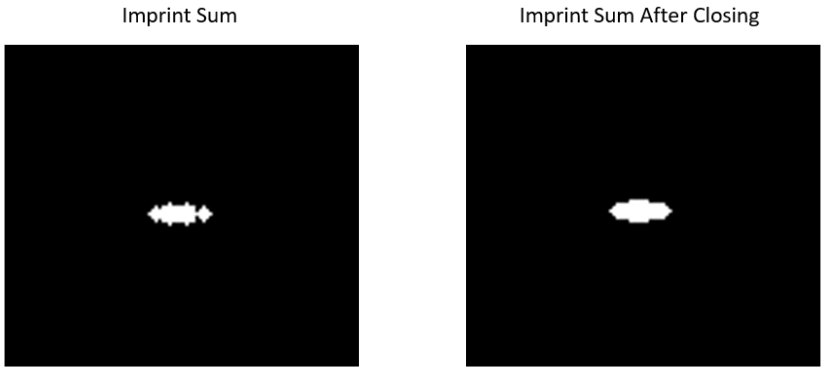


Figure 4.7: Effect of morphological closing on ball imprint area

The out area image is generated by drawing the out-line of the current frame using OpenCV `drawContours()` operation supplying the detected out-line contour. As in squash the area above the bottom of the out-line is 'out' all pixels in this area must be highlighted. The system achieves this by finding the bottom of the out-line in each column of the contour image and setting all values above it to from 0 to 1 as described in Figure 4.8 and uses `numba.jit` to improve the efficiency of the nested loop.

```
for each col in image do
    bottom ← −1
    len ← |col|
    while i < length do
        if col[i] ≠ 0 then
            bottom ← i
            break
        end
    end
    if bottom ≠ −1 then
        below ← [0] × (i − 1)
        above ← [1] × (len − i)
        col ← below + above
    end
end
```

Figure 4.8: Algorithm to convert out-line contour to out area

The system calculates probability of an in/out decision by adding uncertainty to the impact and out area images, using pixel brightness to indicate certainty in the pixel detection. The system uses three values ($\frac{1}{3}$, $\frac{2}{3}$, 1) to indicate detection certainty within an image, which are allocated to the outer-boarder, inner-boarder and centre of the original detected area, as show in Figure 4.9.

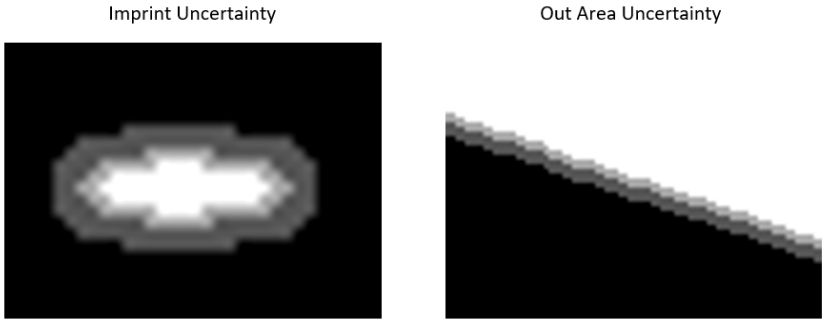


Figure 4.9: Detection probability represented as brightness

The individual images are overlaid using OpenCV `addWeighted()` operation to half value of the pixels in the individual images. When summed the value of each pixel is calculated as displayed in Table 4.1. An overlap of the impact and out area is indicated if a pixel value is greater than the maximum value in a single image, $\frac{1}{2}$, leaving possible overlap indication values ($\frac{2}{3}$, $\frac{5}{6}$, 1).

	None - 0	Outer - $\frac{1}{6}$	Inner - $\frac{1}{3}$	Centre - $\frac{1}{2}$
None - 0	0	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$
Outer - $\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{3}$
Inner - $\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{5}{6}$
Centre - $\frac{1}{2}$	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{5}{6}$	1

Table 4.1: Possible pixel values in imprint and out area weighted sum

Probability is calculated in two steps.

1. The maximum value in the image dictates the upper and lower bound for the probability the ball is out. If the maximum value does not indicate an overlap, by being less than $\frac{1}{2}$, probability is 0.

   Maximum of $\frac{2}{3}$: $0 \geq \mathrm{P} < \frac{2}{3}$

Maximum of $\frac{5}{6}$: $\frac{2}{3} \geq \mathrm{P} < \frac{5}{6}$

Maximum of 1 : $\frac{5}{6} \geq \mathrm{P} \leq 1$

2. Finer granularity within the bounds is calculated using the formula

$$P = b_1 + ((b_2 - b_1) \times r)$$

where b_1 and b_2 are the upper and lower bounds respectively and r is the ratio of the number of pixels at the maximum value to number of pixels in the imprint area great enough to produce the maximum. This step result in a higher probability the more pixels have the maximum overlap value.

4.4 Interface

As the system is designed for ease of use by non-technical squash referee's it is wrapped in a simple graphical user interface made using the built in python library Tkinter shown in Appendix B. The user can load in a video with a button press, using a Tkinter `filedialog` object to open a file explorer window allowing easy file selection. Once selected a button press activates video processing with feedback being given to the user via a playback screen, first displaying the 'collecting data' prompt before playing back the processed video.

The interface window and its components are dynamically sized to be usable on screens of varying resolutions. Also included are buttons to pause and play the video playback process and well as cycling through frame by frame allowing the user to watch back a particular part of the video slowly. Finally an option to display the original video alongside the processed video is given allowing for any obvious detection errors made by the system to be spotted by a user.

The probability of a shot being out is displayed on the processed video as a percentage, if less than 50 the system has classified the shot 'in' displaying in green text, otherwise the shot is 'out' displaying in red text, the decision confidence scale is described in Figure 4.10. An indicator on the display lets the user know when during the playback the ball is detected as making contact with the wall, with the imprint of the ball being drawn in blue as it is generated.

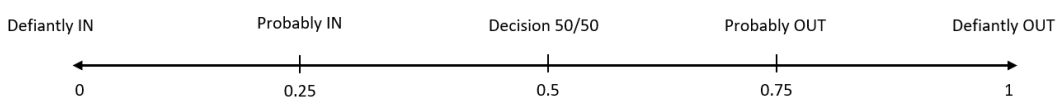


Figure 4.10: How probability value relates to shot in/out decision

Testing

This chapter outlines the strategy and results of tests performed on stages of the systems image processing pipeline. Location in the project archive of test descriptions is highlighted in Appendix D.1.

5.1 Data Set

Firstly a set of video clips of squash shots were required for the system to process. As the system requires high speed camera footage to detect the ball broadcast footage could not be used, requiring the video clips to be recorded manually. Due to the scope of the project a smart phone camera was used, recording at 240fps in 720p, from an elevated position behind the back wall of the court. This position would be accessible at all televised PSA matches due to the current requirement of broadcast cameras.

5.1.1 Data Variation

To ensure the system robustness to reasonable changes in environment that may occur between courts an effort was made to record shots at varying camera locations and orientations as show in Figure 5.1.

Figure 5.1: A sample of camera orientations used in the shot dataset

Furthermore multiple different shot types, exhibiting different behaviour, were recorded. Extreme behaviours were recorded by intentionally hitting the ball:

fast, slow, tight to the wall (skimming), and with other objects moving in the frame. Doing this produced a dataset to cover a larger range of scenario's while taking less time to collect. In total video clips of 74 shots were collected.

5.2 Contact Detection

As the system makes its decision based on the detection data within the contact frames the accuracy with which these frames are selected is fundamental to the accuracy of the decision made. To find the ground truth for each shot I played back the shot frame by frame noting the set of frames in which I observed contact between the ball and the side wall. This process was completed once a day for 3 days to reduce the likelihood of mistakes with the consistently selected frames being chosen as the ground truth for a shot.

The system implementation was run for all 74 shots, with the calculated start and end frame of contact for each shot being recorded in a CSV file. This process was automated via a python script using the `os` library to read the dataset and write the CSV files. Automating this process allowed multiple different implementations to be tested efficiently instead of requiring each shot to be loaded individually. Once all contact frames were detected for the dataset each shots error is calculated using the formula

$$|error| = |s_t - s_d| + |e_t - e_d|$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |error_i|$$

in which s_t and e_t denote the truth start and end frame of contact, s_d and e_d denote the detected start and end frame of contact and n denotes the size of the dataset. The mean absolute error (MAE) is used to evaluate the performance of an implementation, which is simply the average error over all shots in the dataset [23].

Eleven implementations were tested using different trajectory features explained in section 4.2.1 as well as different combinations of threshold values and parameters. Accuracy percentage within different error tolerances was also calculated, the results of these tests can be viewed in Table 5.1. Although implementation 10 showed the best MAE, the accuracy at lower error tolerance's is much higher in implementation 11. Due to the nature of the problem this is of greater significance as an error tolerance exceeding more than two frames would likely effect the reliability of the decision. For this reason implementation 11 was used in the final system producing the contact detection accuracy shown in Figure 5.2.

Test	MAE	Accuracy with frame tolerance				
		0	1	2	3	4
Test 1	3.783	21.62	48.65	71.62	83.78	85.14
Test 2	3.216	43.24	79.73	81.08	83.78	85.14
Test 3	4.284	33.78	66.22	75.68	78.38	81.08
Test 4	4.662	27.03	52.7	67.57	77.03	78.38
Test 5	1.378	47.3	89.19	90.54	93.24	94.59
Test 6	1.608	44.59	71.62	90.54	91.89	94.59
Test 7	1.284	60.81	87.84	90.54	91.89	94.59
Test 8	1.392	50.0	87.84	90.54	91.89	94.59
Test 9	1.635	48.65	71.62	87.84	90.54	91.89
Test 10	1.176	55.41	79.73	91.89	93.24	95.95
Test 11	1.256	60.81	90.54	90.54	91.89	94.59

Table 5.1: Contact detection test results

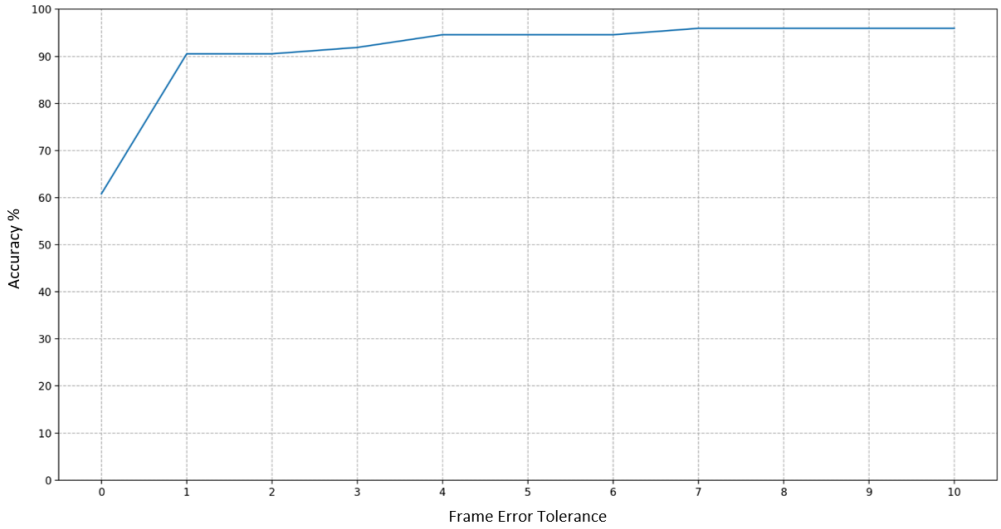


Figure 5.2: Contact detection accuracy of system

5.3 Decision Accuracy

As shot classification is binary, in or out, I instead chose to classify shots by the probability with which it was out, as explained in section 4.4. Each shot's ground truth out probability was calculated as the average decision made by a group of England Squash qualified referee's. Ten referee's viewed each shot once at real-time speed, emulating how they would observe the shot in a match, after which they would classify it as either in or out. The mean average of

each shot was calculated (in = 0, out = 1) to give a final ground truth of out probability between 0-1. This value conveys a level of uncertainty in the referee group which my system can be measured against.

After calculating the aggregated true out probabilities for the dataset I observed that across the 74 shots:

- All referee's agreed in 65: resulting out probability = 0 or 1
- One referee disagreed in 8: resulting out probability = 0.1 or 0.9
- Two referee's disagreed in 1: resulting out probability = 0.2 or 0.8

An automated script similar to the one used in the contact detection tests was used to test multiple implementations efficiently. Allowing thresholds and parameters of the systems probability formula, described in section 4.3.2, to be tweaked as to best model the groups decision uncertainty. The implementation's MAE was calculated as the average difference between the aggregated truth and the system prediction of out probability for each shot in the dataset, shown in Table 5.2.

Test	Agreement %	MAE	Accuracy in prob tolerance			
			0	0.1	0.2	0.3
Test 1	95.95	0.0578	81.08	93.24	94.59	95.59
Test 2	95.95	0.0531	82.43	95.95	95.95	95.95
Test 3	98.65	0.0261	85.14	98.65	98.65	98.65

Table 5.2: Out probability test results

After testing a sample of implementations it was clear that the out probability detection was being bottlenecked by the contact detection accuracy. Due to this, no tweaking of the out probability equation would further improve the decision accuracy. Notably in all of the nine disputed shots the system agreed with the majority in every case. Test 3 showed the best model of the groups uncertainty and was the implementation used in the final system, producing the out probability accuracy shown in Figure 5.3.

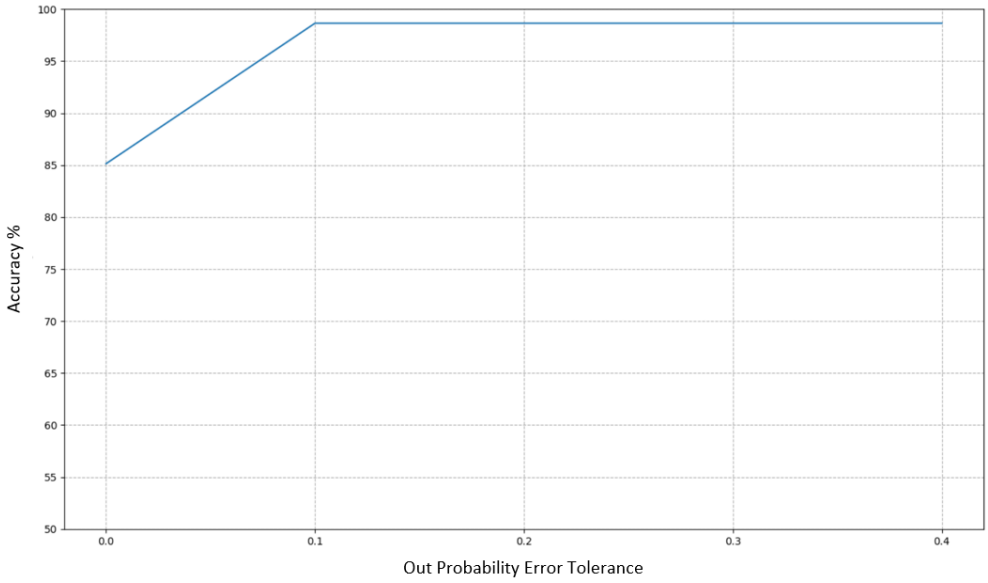


Figure 5.3: Out probability accuracy of system

Evaluation

6.1 System Evaluation

This section evaluates the systems successes and failures against the system requirements specified in section 3.1. The system has been evaluated on the results observed when tested on the shots in the dataset. Due to the national lockdown and subsequent closing of the university squash courts during the COVID-19 outbreak my ability to record shots was impeded, resulting in a smaller dataset than expected [24]. This has limited the quantity of test data producing less reliable results from which to evaluate individual aspects of the system. Table 6.1 summarises the results of the detailed individual evaluations below.

6.1.1 Object Detection

The use of colour thresholding to detect the out-line is extremely consistent due to naturally constricted environment of a squash court. As the colour range used was configured for the out-line colour on the court used to collect data this range would need reconfiguring at different courts. The out-line detection is robust to changes in lighting due to the exaggerated range supplied to the OpenCV thresholding algorithm. The drawback of this large range does result in more unwanted objects being detected at the thresholding stage, but the use of consistent out-line shape descriptors results in the correct object candidate being selected across all shots.

The ball detection process has been observed to not detect the ball if it is very distant from the camera. This is due to size of the ball object in the image being so small that it is indistinguishable from the image noise, resulting in the noise reduction process removing the real ball candidate. However in this case restrictions on ball candidate minimum size will mean the system opts to not classify a ball object instead of potentially misclassifying a ball object. This solution is preferred as gaps in the balls flight can then be predicted by neighbouring detection's.

The ball prediction model uses linear motion assumptions not accounting for the parabolic nature of ball flight, however due to gaps in ball detection being

small the effect of this is negligible. A possible flaw in the current ball prediction model would be seen in the case where the ball makes contact with the side wall on the edge of the out-line while simultaneously being undetected. This would result in the ball position being predicted during contact, as this prediction is linear it could result in the outcome shown in Figure 6.1.

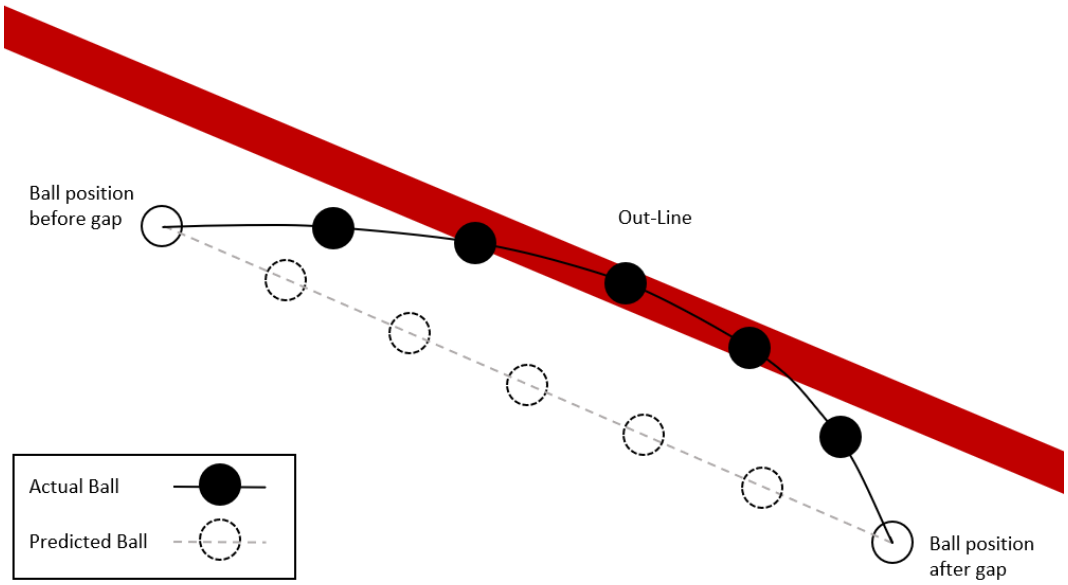


Figure 6.1: Example of shot prediction during contact resulting in incorrect decision

6.1.2 Contact Detection

This process proved difficult due to the restriction of a single camera angle, as the 3D court space had to be interpreted from a single 2D projection. This meant a pattern in the 2D space needed to be found that correlated with the change in 3D space that occurs at the point of contact in the video sequence. Due to the size of the shot dataset this pattern was found by manually inspecting features of trajectory, a larger dataset would have allowed me to utilise machine learning techniques to help find a more consistent detection pattern. Despite this contact was correctly detected, within a single frame, 90.54% of the time.

6.1.3 Out Probability Model

The out probability of a shot calculated by the system was compared to the average decision of a group of ten qualified referees. The aggregated classification of a group is more accurate than an individuals classifications due to the 'wisdom of the crowd' principle [25]. All members of the group were also experts in the field further improving the accuracy of the classification [25]. The system was observed to model the groups decision confidence very well over the dataset, with an mean absolute error of just 0.0261. However the

small dataset meant only a single shot showed significant lack of confidence in the decision, in which eight of ten referee's determined the shot was out. The system models this uncertainty well as the calculated out probability is 0.83, only 0.03 off the group. However due to the dataset's limited size it is unclear if this model would be as accurate across more disputed shots.

The overall decision accuracy of the system is calculated as the percentage of shots from the dataset in which the system agreed with the majority vote from the groups decisions. In this case the system correctly calculates the in/out decision in 73 of 74 shots, producing an accuracy of 98.65%. In total 10 individual referee decisions disagreed with the group across over the 74 shots, evenly distributing these mistakes, over the group size of 10, results in the average individual making an incorrect decision in 1 of 74 shots. This gives the average qualified referee an accuracy of 98.65%, exactly equal to the system accuracy.

In practice the incorrect decisions were not shared completely evenly between the group such that some referee's made more mistakes than others. This means that some of the qualified referee's will have made out-line decisions less accurately than the system.

6.1.4 Performance

The system processes all 74 shots from the dataset in 704.6 seconds, on my personal laptop with an Intel i7 processor, 8GB of memory and a SSD hard drive. This results in a average clip shot processing time of 9.5 seconds, well within the expectation set out in my system requirements. The process time complexity can be described as $O(2nr)$ where n is the number of frames of the video and r is the video resolution.

ID	Achieved	Comments
F1	Yes	The system outputs an in/out decision from a single camera angle input
F2	Yes	Out-line and ball are consistently correctly detected but ball prediction assumptions in fringe cases can be incorrect
F3	Yes	A relatively robust pattern was found to detect contact
F4	Yes	Decision accuracy of 98.65% was equal to the average referee, but greater than some individual referee's
F5	Yes	Out probability is displayed in the processed shot playback
NF1	Yes	Average shot processing time was 9.5s
NF2	Yes	A simple GUI with intuitive controls wraps the system

Table 6.1: Evaluation of system requirements

6.2 Comparative Evaluation

The system created in this project can be compared to existing line calling systems in other sports, in most cases these use a synchronised multi-camera approach [4]. As the Hawk-Eye tennis electronic line calling system aims to achieve the objective most similar to my system I will compare the difference and similarities between them in more depth.

6.2.1 Process Comparison

Ball Detection

The Hawk-Eye system first detects the centre of the ball in the 2D space (x, y) of a frame from each of the ten pre calibrated cameras surrounding the court. This allows triangulation to be used for predicting the 3D (x, y, z) position of the ball [8] [5]. The limitation of a single camera means my system can not triangulate a 3D position, however initial step of detecting the ball centre in 2D is achieved by my system.

Ball Tracking

Both mine and the Hawk-Eye system produce a ball track from the balls position over time through a video sequence. In the case of Hawk-Eye this shows the ball in 4D (x, y, z, t) compared to my systems (x, y, t) representation. To handle occlusion and determine true ball objects from noise Hawk-Eye uses a flight model that accounts for air resistance and rotation [26]. My system flight model does not take into account these factors, however they have less impact in the game of squash due to the comparatively short shot travel distance and smaller smoother ball.

Impact Calculation

Both systems calculate the impact area/ball footprint from the trajectory of the ball, instead of measuring it directly. In both system models the angle of impact and speed of the ball are considered while also incorporating maximum contact diameter threshold restrictions [27] [5]. However the lack of a full 3D position of the ball in my system means ball depth is not directly recorded introducing larger error when predicting the time of contact in my system.

Outcome

The Hawk-Eye system differs from my system in that it returns a binary in/out decision, compared to my continuous 'out probability'. In doing do it may "inadvertently mislead the public about the degree of certainty that be brought to a scientific measurement" [28]. It is suggested that "incorporating information about measurement error into the presentation" would remove these

concerns [28]. This approach is used in my system via a decision confidence score highlighting to the user that decision made by the system is not without error.

6.2.2 Accuracy Comparison

The Hawk-Eye system's is measured by it's ball detection accuracy. This is stated as having a mean error of 2.6mm when compared to a high speed camera on the playing surface [5]. I was unable to test my systems accuracy in this way, making direct comparisons between system ball detection accuracy impossible. However we can compare the systems by their improvement over their human alternative when making out-line decisions.

In tennis it is predicted that "8.2% of all line calls involving balls within 100mm of a court line will be called incorrectly by line judges" [29]. The Hawk-Eye system, given its detection error margin of 2.6mm, would make the wrong decision in half of the shots within this margin of the line. If the shots were to be evenly distributed over the 100mm distance from the line the system would have an incorrect decision probability of 1.3%, resulting in an accuracy 6.3 times greater than the line judges for shots within 100mm of the line.

From my dataset of the four shots visibly close to the out-line boundary I found that a referee would make the incorrect decision 10% of the time, a similar human error rate to that of tennis. In all these shots my system returned the correct decision, however due to the small sample a reliable system accuracy for these cases can not be determined.

6.2.3 Cost Comparison

Implementation of the Hawk-Eye system at the Farmer's Classic tournament in Los Angeles was estimated at $60,000 - $70,000 per court [30]. This is high due to the expensive camera equipment and its installation required for the system to operate. Furthermore each camera must be precisely calibrated as their relative locations must be known for the ball's 3D position to be triangulated.

My system, being built off a single camera feed, does not have any of these attached costs. Furthermore as triangulation is not used camera calibration is not required reducing the installation costs. The equipment requirements of a single 240fps camera and consumer grade laptop allow the cost of system implementation to be well below $1,000. Costing less than 2% of the Hawk-Eye system will allow lower level divisions and national leagues with a smaller budget to afford the system implementation. This is often a major argument against the Hawk-Eye system [4].

6.3 Project Management

At the beginning of the project a Gantt chart was created to estimate the time required for different stages of the project development and schedule them accordingly. Throughout the project implementation stage, starting in late January, a record of the work achieved each week was kept. This allowed another Gantt chart of the actual work schedule to be made after the implementation was finished in April, both of which can be found in Appendix A.

The main changes to the original schedule were due to the collection of a second set of data. The larger more varied selection of shots highlighted problems with the systems methods of object detection requiring more time to be spent in this area than originally planned. This had a knock on effect causing the subsequent ball tracking tasks to be pushed back by two weeks. The time was then made up in the decision making tasks which were completed in a shorter time than expected. The choice to create the simple GUI at the start of the implementation process proved useful in visualising how the system was working and the spotting of bugs easier. During the project the use of automated testing scripts for contact detection and decision making made testing faster and easier, despite the overhead required to develop them.

Code and test results for the project were stored in a personal GitHub repository. Frequent commits meant in the event my laptop was lost or damaged I would not lose any of the progress I had made. The repositories commit graph shows how the implementation workload was well spread over the three month period, shown in Appendix A.3.

Due to the unexpected incident of the international COVID-19 outbreak I was unable to expand my robustness testing further using a larger dataset as I had initially planned. This resulted in the final system tests being performed on a smaller than expected dataset. In hindsight a large dataset should have been collected at the start of the project.

Conclusion

7.1 Project Goals

Although the project was limited by the size of the dataset, not allowing strong conclusions to be made of edge case shot decisions, the overall project goal of creating a computer vision system to determine out-line decisions in squash has been achieved. This system was built with the intention of being utilised in the decision review process of squash matches on the PSA world tour. Although unable to test this system in a real-world scenario I believe the work achieved throughout the project could be easily transferred into a system compatible with the current set-up at a PSA world tour match. I also believe the option to use this system for player reviews would be widely appreciated by the current players/commentators on the tour, as highlighted in their comments in section 1.1.

Furthermore, original discoveries where made in the area of 3D trajectory analysis from a 2D space, in which different engineered features were examined to find contact patterns. These features can be reused in different projects using a single camera input to calculate behaviours of 3D trajectory. This meets my secondary project goal to investigate the effectiveness of different image processing operations on the task of ball and line detection.

7.2 Future Work

As previously highlighted a larger dataset containing more edge case decisions would be collected to improve the reliability of the system accuracy test results. This would be required if the system was to be adopted into professional sport. This would also provide more data from which to develop the contact detection processes which currently bottlenecks the system accuracy. Machine learning techniques could then be explored to find more consistent contact patterns in this dataset.

Further development of the system to automate the configuration of various threshold and parameters required to detect the ball and out-line, such as the colour range. Automating this process would allow the system to be used at any court independent of out-line, ball or wall colour as well as video resolution.

Further development into using a ball position prediction model, as to account for the incoming trajectories parabolic motion, will make the system more accurate as features derived from predicted ball position are used to calculate contact point which in turn is used to calculate the in/out decision. This will also help to resolve the theoretical error in the scenario described in Figure 6.1.

Main body word count: 8285

References

[1] US Squash. *Squash Facts*. 2015. URL: https://www.ussquash.com/squash-facts/.

[2] Mark Nadolny. *Shuttlecock and balls: The fastest moving objects in sport*. 2014. URL: https://olympic.ca/2014/09/11/shuttlecock-and-balls-the-fastest-moving-objects-in-sport/.

[3] WSF. "Specifications for Squash Courts". In: *World Squash* (2016).

[4] Graham Thomas, Rikke Gade, Thomas B. Moeslund, et al. "Computer vision for sports: Current applications and research topics". In: *Computer Vision and Image Understanding* 159.April (2017), pp. 3–18. ISSN: 1090235X. DOI: 10.1016/j.cviu.2017.04.011.

[5] Paresh R. Kamble, Avinash G. Keskar, and Kishor M. Bhurchandi. "Ball tracking in sports: a survey". In: *Artificial Intelligence Review* 52 (2019), pp. 1655–1705. ISSN: 15737462. DOI: 10.1007/s10462-017-9582-2.

[6] Cheng Zhang, Fan Yang, Gang Li, et al. "MV-Sports: A Motion and Vision Sensor Integration-Based Sports Analysis System". In: *Proceedings - IEEE INFOCOM* 2018-April (2018), pp. 1070–1078. ISSN: 0743166X. DOI: 10.1109/INFOCOM.2018.8485910.

[7] N. Owens, C. Harris, and C Stennett. "Hawk-Eye tennis system". In: (2005), pp. 182–185. DOI: 10.1049/cp:20030517.

[8] Hawk-Eye Innovations. "Electronic Line Calling Technology: How it works". In: (2016).

[9] Babette M. Pluim. "The evolution and impact of science in tennis: Eight advances for performance and health". In: *British Journal of Sports Medicine* 48 (2014), pp. 3–5. ISSN: 14730480. DOI: 10.1136/bjsports-2014-093434.

[10] Rupesh Kumar Rout. "A Survey on Object Detection and Tracking Algorithms". In: (2013), p. 75.

[11] Rupali S Rakibe and Bharati D Patil. "Background Subtraction Algorithm Based Human Motion Detection". In: *International Journal of Scientific and Research Publications* 3.5 (2013), pp. 3–6. URL: http://www.ijsrp.org/research-paper-0513/ijsrp-p17106.pdf.

[12] M Sankari and C. Meena. "Estimation of Dynamic Background and Object Detection in Noisy Visual Surveillance". In: *International Journal of Advanced Computer Science and Applications* 2.6 (2011). ISSN: 2158107X. DOI: 10.14569/ijacsa.2011.020611.

[13] Yu Huang, Joan Llach, and Chao Zhang. "A method of small object detection and tracking based on particle filters". In: *International Conference on Pattern Recognition* (2008). ISSN: 10514651. DOI: 10.1109/icpr.2008.4761480.

[14] Nilima Kulkarni. "Color Thresholding Method for Image Segmentation of Natural Images". In: *International Journal of Image, Graphics and Signal Processing* 4.1 (2012), pp. 28–34. ISSN: 20749074. DOI: 10.5815/ijigsp.2012.01.04.

[15] Gareth J. Lewis, J. Cris Arnold, and Iwan W. Griffiths. "The dynamic behavior of squash balls". In: *American Journal of Physics* 79.3 (2011), pp. 291–296. ISSN: 0002-9505. DOI: 10.1119/1.3531971.

[16] Saumil Sachdeva. "Detection and Tracking of a Fast-Moving Object in Squash using a Low-Cost Approach". PhD thesis. Delft University of Technology, 2019.

[17] Olli-Pekka Heinisuo. *OpenCV Python*. URL: https://pypi.org/project/opencv-python/.

[18] Python Software Foundation. *Graphical User Interfaces with Tk*. 2019. DOI: 10.1017/9781108591942.019. URL: https://docs.python.org/3/library/tk.html.

[19] P. KaewTraKulPong and R. Bowden. "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection". In: *Video-Based Surveillance Systems* (2002), pp. 135–144. DOI: 10.1007/978-1-4615-0913-4_11.

[20] Dovzhyk Mykhailo. *Angle between two vectors*. URL: https://onlinemschool.com/math/library/vector/angl/.

[21] Linus J Dowell and Gary Krebs. "A formula for comparison of selected sport ball compressibility". In: 25.1 (1991), pp. 34–37.

[22] Claire Davis and Martin Strangwood. *Squash ball bounce*. URL: https://www.youtube.com/watch?v=5IOvqCHTS7o.

[23] Cort J. Willmott and Kenji Matsuura. "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". In: *Climate Research* 30.1 (2005), pp. 79–82. ISSN: 0936577X. DOI: 10.3354/cr030079.

[24] University of Southampton. *COVID-19 (coronavirus): information and guidance*. URL: https://www.southampton.ac.uk/news/statements/coronavirus.page.

[25] Camelia Simoiu, Chiraag Sumanth, Alok Mysore, et al. "Studying the "Wisdom of Crowds" at Scale". In: *Proceedings of the Seventh AAAI Conference on Human Computation and Crowdsourcing (HCOMP-19)* Volume 7 No 1 (2019), pp. 171–179. URL: http://web.stanford.edu/%7B~%7Dcsimoiu/doc/wisdom-of-crowds.pdf.

[26] Feng Li, Lu Liu, Qiaohui Wang, et al. "Tennis balls judgment model based on numerical simulation". In: 01018 (2017), pp. 2016–2018.

[27] Rod Cross. "The footprint of a tennis ball". In: June (2017). DOI: 10.1007/s12283-014-0159-x.

[28] Harry Collins, Robert Evans, Harry Collins, et al. "You cannot be serious ! Public understanding of technology with special reference to " Hawk-Eye "". In: (2011). DOI: 10.1177/0963662508093370.

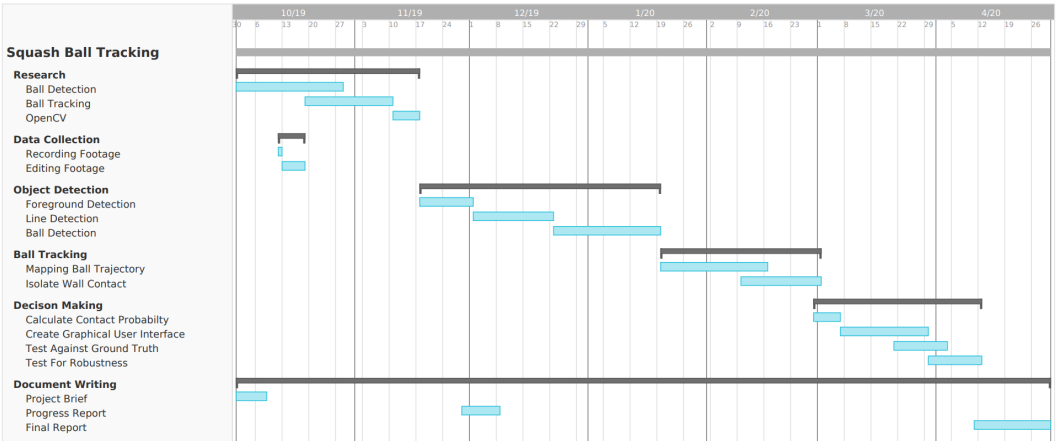
[29] George Mather. "Perceptual uncertainty and line-call challenges in professional tennis". In: April (2008), pp. 1645–1651. DOI: 10.1098/rspb.2008.0211.

[30] Simon Barton. *Unlocking Hawk Eye: What it means for tennis, the ATP, WTA and ITF*. URL: https://channels.theinnovationenterprise.com/articles/unlocking-hawk-eye-what-it-means-for-tennis-the-atp-wta-and-itf.

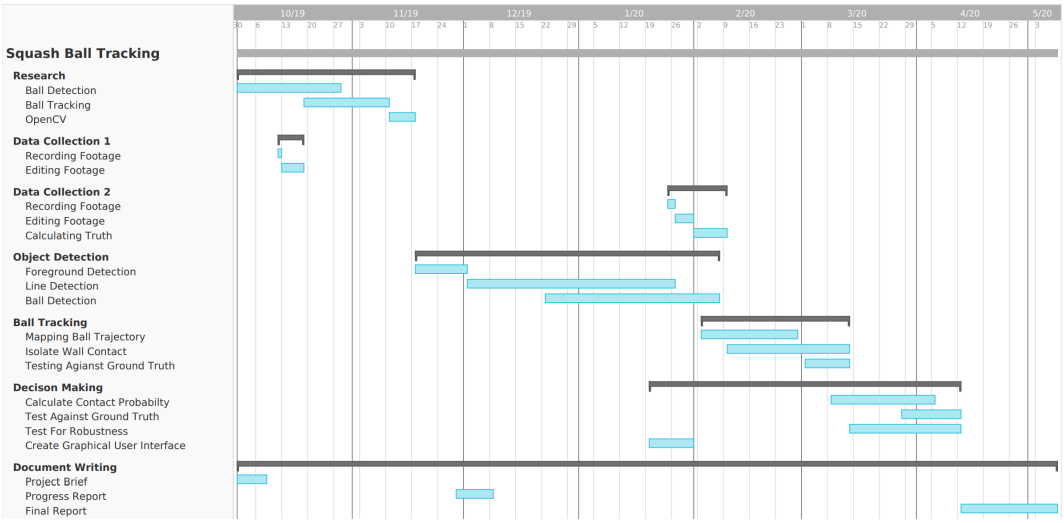
Appendices

Appendix A

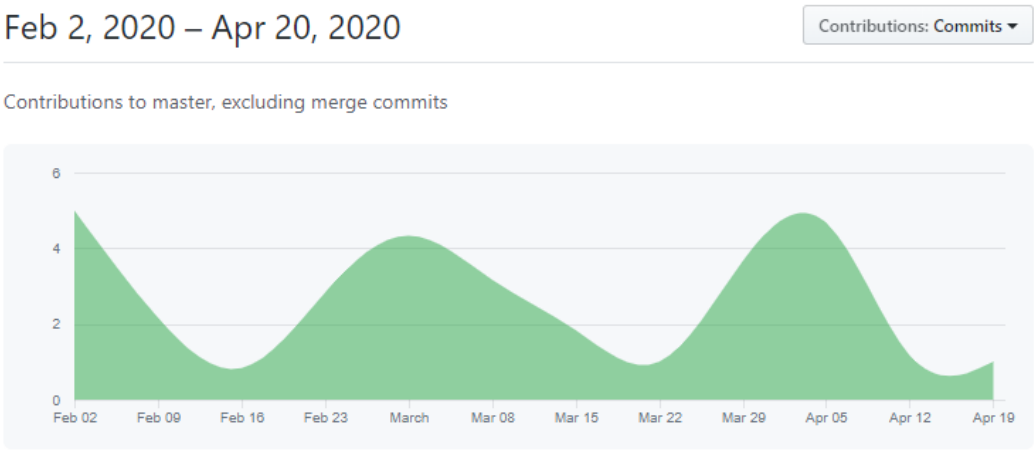
A.1 Initial Gantt Chart



A.2 Final Gantt Chart

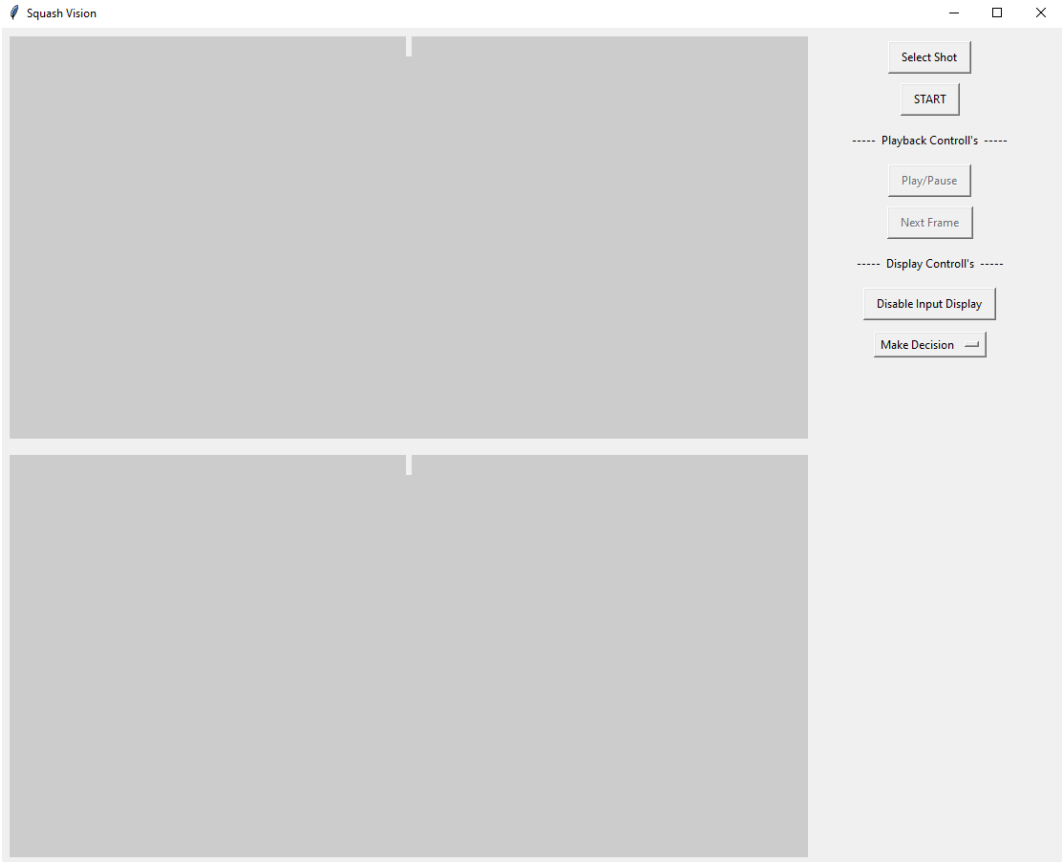


A.3 GitHub Commits

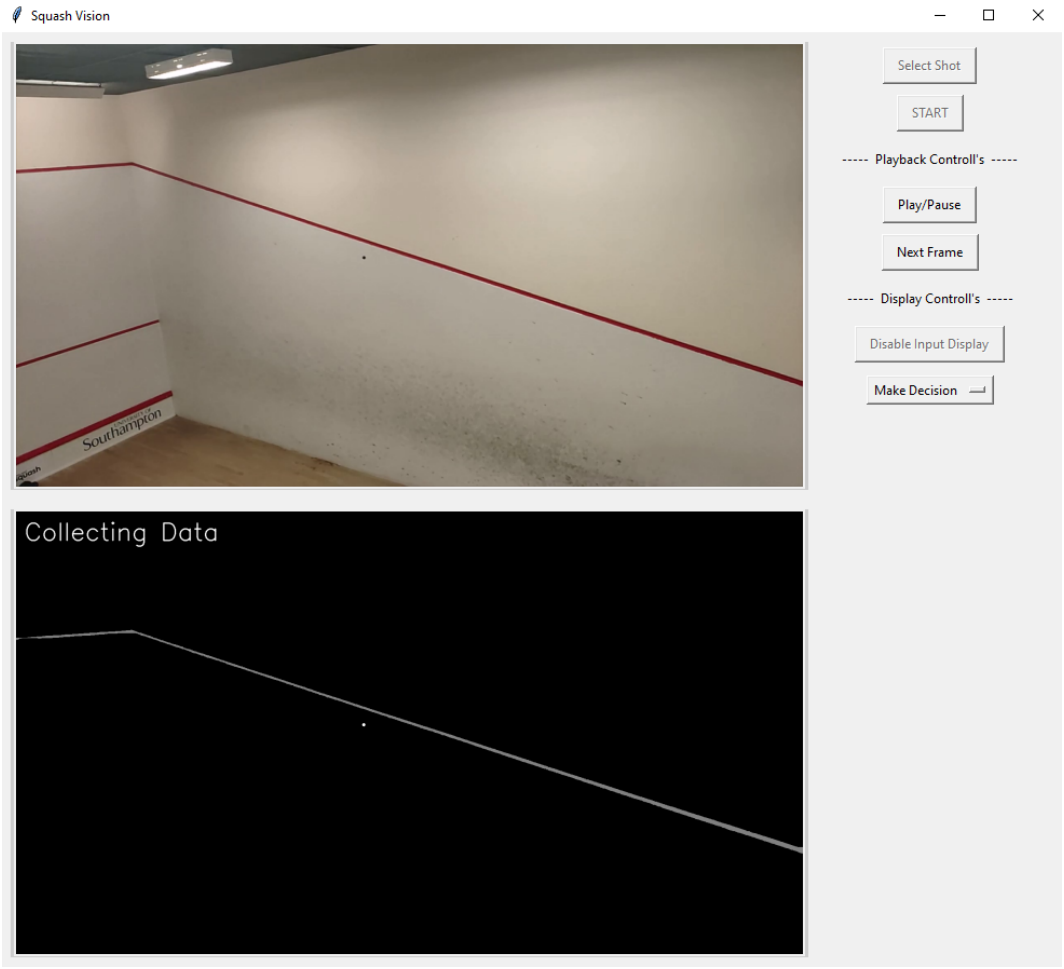


Appendix B

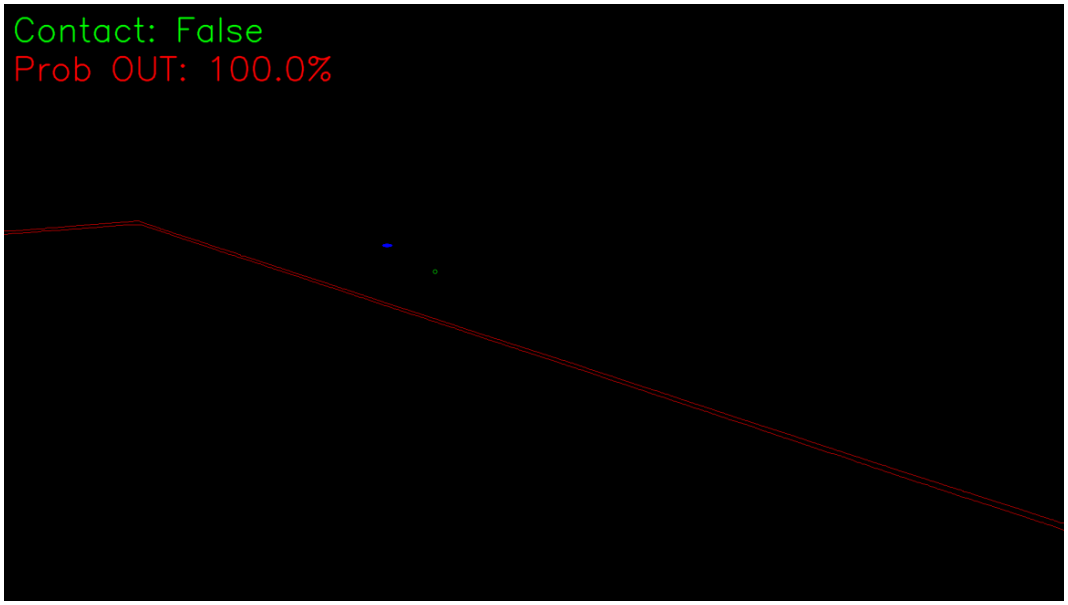
B.1 Application Interface: Initial

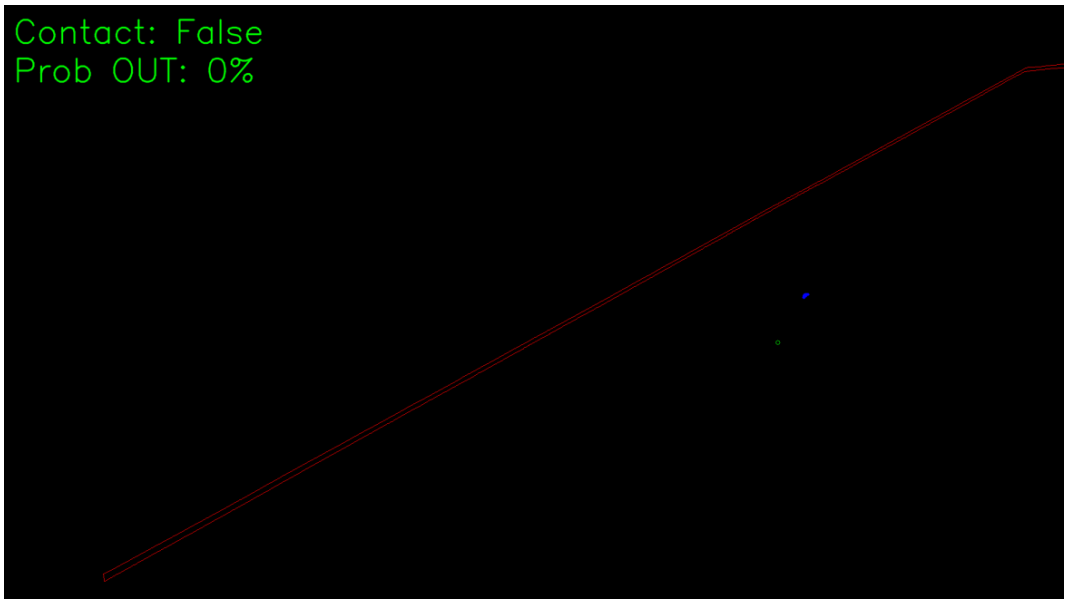


B.2 Application Interface: Processing



B.3 Application Interface: Decision





Appendix C

C.1 Participant Information Form

Participant Information

Ethics reference number: **ERGO**/FEPS/**41626.A1**	Version: 1	Date:3/09/2019
Study Title: Ball Tracking to Determine Out-Line Decisions in Squash		
Investigator: **Luke Gibson**		

Please read this information carefully before deciding to take part in this research. If you are happy to participate you will be asked to indicate your consent to take part either verbally or by selection if an online questionnaire. Your participation is completely voluntary.

What is the research about? This research project is part of the COMP3200 project. The research will only involve :

Questionnaire

Why have I been chosen? You have been approached because you are known to the student(s) or because you have been identified by the students as being appropriate for the research

What will happen to me if I take part? You will take part in a short

Questionnaire

Are there any benefits in my taking part? The study will add to current knowledge, as well as being valuable practical learning for the COMP3200 project student

Are there any risks involved? No sensitive issues will be discussed and there are no risks beyond that which would normally be experienced in everyday life.

Will my data be confidential? Please do not give any identifiable personal information. The project student will retain anonymous data until the end of the Project. No video or audio recording will occur **What happens if I change my mind?** You may withdraw at any time and for any reason. You may decline to give your consent and not take part in the study without penalty.

What happens if something goes wrong? If you have any concern or complaint, contact Luke Gibson lg1n17@soton.ac.uk or Dr Sasan Mahmoodi (sm3@ecs.soton.ac.uk), otherwise please contact Research Governance Office (02380595686, Rgoinfo@soton.ac.uk).

C.2 Participant Consent Form

Consent Form

Ethics reference number: **ERGO**/FEPS/**41626.A1**	Version: 1	Date: 03/09/2019
Study Title: Ball Tracking to Determine Out-Line Decisions in Squash		
Investigator: **Luke Gibson**		

Please read the following and indicate verbally
if you agree with the following statements:

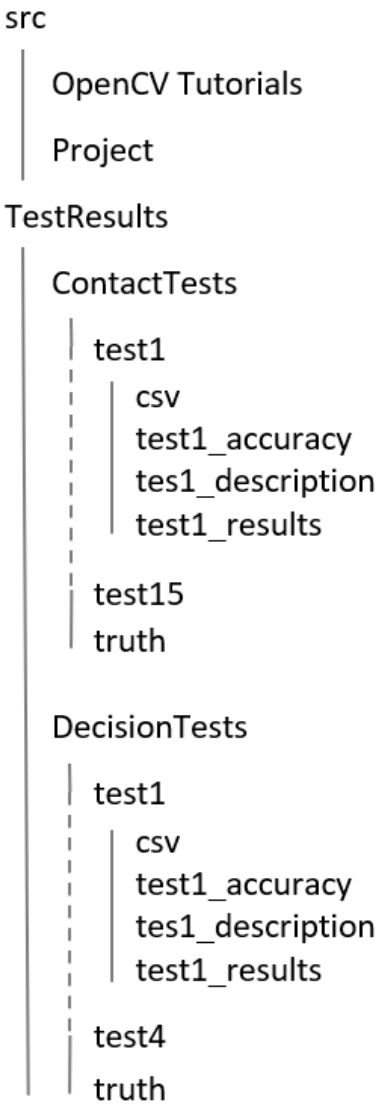
I have read and understood the Participant Information and have had the opportunity to ask questions about the study.

I agree to take part in this study.

I understand my participation is voluntary and I may withdraw at any time and for any reason.

Appendix D

D.1 Project Archive Contents



D.2 Original Project Brief

Ball Tracking to Determine Out-Line Decisions in Squash

Luke Gibson
Dr Sasan Mahmoodi

4th October 2019

1 Problem

In the game of squash if a player hits the ball and it makes contact on, or above, the out-line the player loses the point. The out-line is horizontal on the front and back wall but angled on both side walls. This, coupled with the referee usually being located centrally behind the back wall, means that they have a poor viewing angle when determining if a ball was above or below the side wall out-line. The inconsistent and unreliable decisions made can be the difference in a player winning or losing a match, and at the professional level cost players £10,000's.

2 Goals

I aim to write a program that given a single video feed per wall will calculate whether the ball was below or above the out-line at the point of contact with the wall. There by creating a more reliable way for referee's to decide on the outcome of in/out line calls.

3 Scope

To achieve this I intend to use computer vision techniques to track the ball in recorded footage, then utilising machine learning technologies map the path of the ball detecting the instance in which the ball made contact with the wall. I plan to use physical factors such as angle of ball flight and ball deformation to detect between which frames contact was made then extrapolating previous known ball points to fill in the path between frames. This would allow me to predict where the ball made contact in the event it was not captured in an individual frame due to the ball speed and limited camera frame rate due to hardware expense.