MSc in Embedded Systems

Detection and Tracking of a Fast-Moving Object in Squash using a Low-Cost Approach

Saumil Sachdeva

Q&CE-CE-MS-2019-09

2019

TU Delft

ALTEN

DETECTION AND TRACKING OF A FAST-MOVING OBJECT IN SQUASH USING A LOW-COST APPROACH

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Embedded Systems

by

Saumil Sachdeva

July 2019

Saumil Sachdeva: *Detection and Tracking of a Fast-Moving Object in Squash using a low-cost approach* (2019)

MSc Thesis number: Q&CE-CE-MS-2019-09

The work in this thesis was made in the:

Quantum and Computer Engineering Group
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

In collaboration with:

Technical Software West
Alten Technology
Alten Nederland
Capelle aan den IJssel

Committee Chair:	Dr. ir. J.S.S.M. Wong (CE QCE EEMCS)
Supervisors:	Dr. ir. A.J. van Genderen (CE QCE EEMCS)
	Dr. Ir. D. Blom (Alten Nederland)
External Committee Members:	Dr. M. Zuniga (E&NS ST EEMCS)
	Dr. A. Prins (Alten Nederland)

ABSTRACT

Advancement in technology has given rise to the need for technology to be utilized in extracting meaningful game-play information from sports. To do so in a ball-game like squash, the prime objective is to perform ball-tracking in an adequate and efficient manner. In squash, ball-tracking is complex due to the small size of the ball, the high-speed movement and constant occlusion due to the continuous movement of the players. The current state-of-the-art ball tracking methods use high-speed cameras along with high-computation power resources to solve these problems in similar sports such as tennis. The aim of this thesis is to solve the challenges in ball-tracking for squash using a low-cost approach with low-computation power resources and a single camera view.

A ball-detection system with a high-accuracy and a ball-tracking system which can optimally tackle the problem of occlusion is developed using computer-vision techniques and by utilizing the cues from the game itself. The implementation is carried out on a Raspberry-Pi which is characterized as a low-computation platform with an Arm Cortex-A53 processor. The results show that the tracking-problem can be solved using a low-cost approach for the challenging scenarios that are present in squash. The 2D trajectory of the ball generated as a result can be used for various applications such as line-calling, shot analysis and game analysis.

Keywords: Squash, Ball-Tracking, Low-Cost, Computer Vision

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Alten Nederland for providing me with such a challenging yet an interesting and fun graduation project. A special thanks goes to Dr. David Blom for his earnest supervision and for pushing me in the right direction whenever I was stuck in any problem. A big thanks to my technical manager Dr. André Prins for offering such an interesting project and for his canny ideas during the project. Also, a huge thanks to my business manager Arnold Schutter for taking such good care of me during my time at the office and for giving me the flexibility I needed during my work.

I would like to thank Dr. Arjan van Genderen for his patience and for his supportive supervision during the project. Thank you for accepting me under your supervision and for always being present to answer all my questions. This thesis wouldn't have been possible without your support and your invaluable inputs during the project.

A huge thanks to Dr. Stephan Wong for accepting me in the Computer Engineering research group for my thesis and to Dr. Marco Zuniga for agreeing to be a part of my thesis committee.

A big thanks to my family back in India, who made sure that I didn't get carried away on the good days and didn't become miserable during the bad days.

Last but not the least, a big shoutout to all my friends for their continuous support to make sure that I was constantly giving my best so that I could complete my thesis in a timely manner and celebrate with them afterwards!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

1.1 MOTIVATION

Squash is termed as the healthiest sport in the world Forbes (2003) as it is said to help increase the balance, agility and strength of a player. It is quickly gaining popularity, with an increasing number of players joining the professional squash circuit along with many amateur players participating in the game to keep themselves fit and healthy. With such a high number of players starting to play this game, there is a need to create game-play information so that the players can improve their game and analyze their shot patterns.

To create such a system, the squash ball needs to be tracked efficiently and accurately throughout the match. There are already similar systems available for the popular ball-games such as tennis and cricket which are utilized in professional matches to aid a referee/umpire in making a correct decision. The most popular of such systems, the **Hawk-Eye System**, consists of multiple high-speed cameras with a number of high-computing resources to accurately map the 3D trajectory of the ball in those games (Owens et al., 2003). The ball trajectory is used in tennis for the purpose of line-calling. In it, the system can be used by the players to review whether the tennis ball lands inside or outside of the court boundary during close-calls. In cricket, the ball trajectory can be used by the players to appeal for a review of the LBW (Leg Before the Wicket) decision made by the umpire. The trajectory is also used in game-analysis of a player and for enhancing the sport-broadcasts. There are similar solutions available for amateur tennis players as well, such as In/Out (2017). Currently, there are no such solutions available for either professional or amateur squash games.

In view of this fact and the growing popularity of the sport, it is important to create a ball-tracking system for the game of squash. The process of detection and tracking in squash is quite challenging due to various factors such as the small size of the ball, its high-speed movement, the constant player occlusion, and real-time constraints. Moreover, ball-tracking in sports has always been done utilizing high-computation power, but since squash lacks in mainstream popularity as compared to sports such as cricket or tennis, the prudent approach is to create a low-cost solution for squash. This low-cost approach utilizes a micro-processor and a single camera-view to perform ball-tracking. Having a low-cost solution is advantageous since it is a realistic approach for ball-tracking in the scenario of squash and as it also presents an interesting research-challenge to perform the intricate ball-tracking process in a novel way.

(a) Hawk-Eye in Cricket **(b)** Hawk-Eye in Tennis

Figure 1.1: Hawk-Eye System for Ball Tracking Hawk-Eye Innovations (2019)

1.2 PROBLEM STATEMENT AND THESIS GOALS

The need for creating a ball-tracking system in squash as discussed in Section 1.1 translates to the following problem-statement for the thesis:

Detect and Track a Squash ball using a single camera view and implement the system on a low-computation platform

The squash-ball represents the category of fast-moving-objects (FMO) as described in Rozumnyi et al. (2017) where the author defines a FMO as "an object that moves over a distance exceeding its size within the exposure time." Since, ball-tracking in squash involves scenarios with high-occlusion, this project aims to perform detection and tracking of a FMO under occlusion through squash.

This problem expands to these specific and measurable goals and objectives for the project:

- Study the ball-detection methods available in the literature for ball-games and identify the methods which are suitable for detecting the small-sized and fast-moving squash ball.
- Implement a ball-detection system which can optimally detect the squash ball. Measure and analyze the accuracy of the system for different squash-matches as datasets.
- Study the ball-tracking methods available in the literature for ball-games and identify the methods which can be implemented efficiently on low-cost platforms and provide good results under occlusion conditions.
- Implement the suitable ball-tracking method(s) using a single-camera view and on a Raspberry Pi 3 which acts as a low-cost platform. Compare the trajectories generated from these methods to the actual detections to analyze the performance of the tracking-methods.
- Analyze the optimization steps that can be taken for carrying out the complete process on a Raspberry Pi 3. Measure and analyze the results of the steps using a timing analysis.
- Evaluate whether the process can be optimized to operate under real-time constraints using a timing-analysis. Using the results, assess the applications of the system.

1.3 METHODOLOGY

The problem of ball-tracking has been extensively researched, with various algorithms and techniques producing good results in different sports (Kamble et al., 2017). The ball-tracking process is a systematized process which consists of four main stages irrespective of the sport it is being applied to. This can be observed in Figure 1.2.

The first stage of the process is the calibration stage where the intrinsic and extrinsic camera parameters are calculated after placing the camera at the back of the court. In this step, the court boundaries are measured and the size of the ball is related to a size in pixel values using the camera parameters. This step is not performed for this thesis, instead, videos of professional squash matches are used as the datasets (Section 3.1). The coordinates of the court boundaries have been pre-calculated for the videos and the size-thresholds have been calculated using a statistical analysis, as will be discussed in Chapter 4.

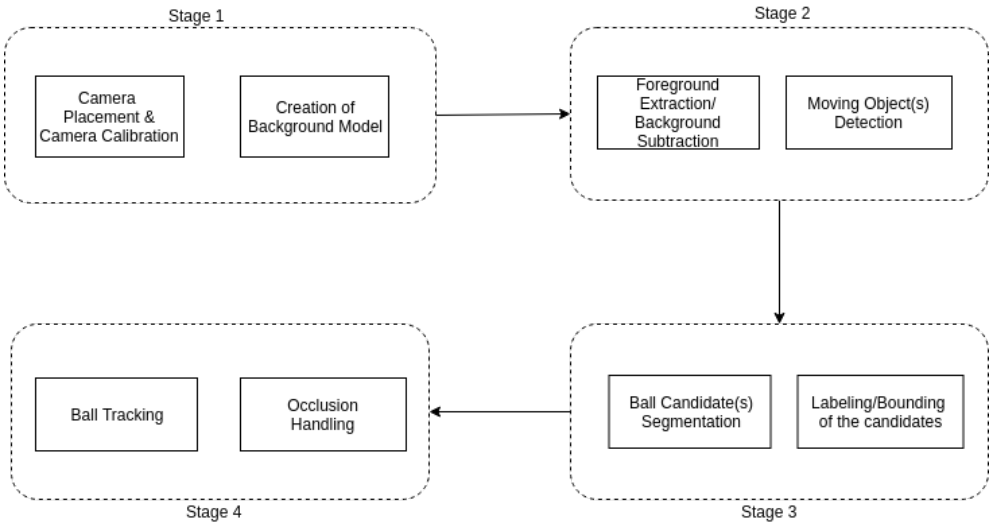


Figure 1.2: Ball Tracking-Process Pipeline

In the second stage, the input image frames are pre-processed by applying certain image processing operations on the image-frames. First, the foreground of the image is separated from the background (Section 3.2). This is followed by operations to prepare the image-frames to be processed for the detection stage (Section 3.3 - Section 3.4).

In the third stage, the ball is detected (in each frame) based on the property of the ball and the cues from the game itself (Chapter 4). The detection methods differ for every sport as they focus on the specific problems of those sports.

In the final stage, the detections in each frame are combined together to form a coherent ball trajectory by solving the problem of mis-detections (occlusion) and multiple-detections (over-segmentation). The output of this stage is a smooth final ball-trajectory (Chapter 5).

Squash as a sport, bears a lot of similarity to tennis. Both sports contain two players, a single fast-moving ball and a limited playing region. Therefore, ball-tracking in tennis is used as the focal-point to investigate detection and tracking methods in the literature. Other sports are also used for reference to solve the challenges that cannot be established in tennis such as the small size of the ball and the constant occlusion by the players.

To create a working prototype solution for measuring the results, the open source computer vision library (Bradski, 2000) is utilized for the various computer vision applications in the project. The implementation is done using Python3 on a Raspberry-Pi 3B+ computer which has a 1.4GHz 64-bit Arm Cortex A-53 CPU and 1GB of Memory.

1.4 THESIS OUTLINE

The thesis is outlined in the following manner:

In Chapter 2, the related work regarding ball-tracking is discussed. As explained above, the primary focus is kept at studying works for the game of tennis. For specific challenges such as occlusion and non-linear motion, works from other sports and domains are also reviewed. The chapter is concluded by reevaluating the research goals specified in Section 1.2 and establishing the hypotheses regarding the research goals.

Chapter 3 discusses the pre-processing stage. It covers the image processing operations performed on the input images and the process of foreground extraction in those images. The process of obtaining the dataset is also discussed in this chapter.

In Chapter 4, the ball-detection step is explained. The steps that give the best results in the specific scenario of squash are reviewed and implemented. Furthermore, their results are discussed and the chapter is concluded with a final remark comparing the methods as hypothesized from the literature to the methods that are implemented.

Chapter 5 discusses the ball-tracking stage in detail. The tracking methods that have been implemented are explained and compared. The results of these methods are evaluated by a 2D ball trajectory as generated in a candidate feature image in Yu et al. (2004).

In Chapter 6, the steps taken to optimize and run the process on the Raspberry-Pi are reviewed. The results of these steps are discussed and evaluated through a timing analysis of the process.

Finally, Chapter 7 concludes the thesis with a summary of the ball-tracking process. The hypothesis established at the end of Chapter 2 are evaluated with the implementation. Furthermore, the main contributions of the work are listed and the future directions of this work are evaluated.

2 RELATED WORK

The literature review carried out to understand the ball-tracking process is discussed in detail in this chapter. Due to the organized nature of the whole tracking process, the literature review is divided into sections pertaining to each step of the tracking process. After the literature review, the chapter is concluded by forming the hypotheses regarding the goals and objectives of the thesis based on the information from the literature.

2.1 PRE-PROCESSING

The pre-processing step generally involves decisions regarding the camera setup and the process of separating the foreground of the image which consists of moving objects in a frame from the background which contains static, non-moving objects. These steps are further discussed in the following subsections.

2.1.1 Camera Setup

The camera setup is a crucial part of any ball-detection system. A high image quality camera with a clear field of view contributes to a smooth detection process. In this regard, the placement of the cameras has to be in such a way that it covers the whole playing field and does not interfere with the game in any way. Many times, along with the ball, other objects of interest such as the players and the court lines also need to be detected along with the ball, and the camera setup has to take this into account as well.

Single Camera Approach

A single camera is used in a setup for ball-tracking due to the ease of use. It is used in Polceanu et al. (2018), where a single wide angle fish-eye lens camera for a Raspberry Pi is used to record the videos. In Yan et al. (2005), a single-camera view from behind the tennis-court is used. In Qazi et al. (2015), single-camera is mounted on a quadcopter to record the videos.

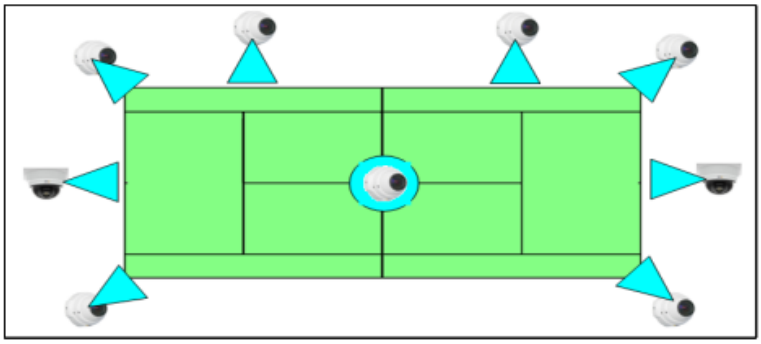


Figure 2.1: Multicamera set-up in Conaire et al. (2009)

Multi-Camera Approach

Multi-camera approaches are generally used to handle occlusion efficiently or to map larger areas. Although these approaches provide a significant advantage in localizing the position of the ball, their setup is complex as compared to the single-camera approaches.

In Owens et al. (2003), they use multiple high-performance cameras that are set up in the tennis court for broadcasting purposes. In Conaire et al. (2009), a camera setup of nine IP cameras is utilized which also includes one overhead camera (Figure 2.1). This setup covers the entire court region from every corner of the court.

A camera setup involving six cameras is utilized in Pingali et al. (2000), in which four cameras are placed on the sides of the tennis court and two behind. In it, there are more cameras on the sides of the court as compared to Conaire et al. (2009), which helps in localizing the ball better but has to also take into account the noise generated from the sides of the court. The system in Fazio et al. (2018) utilizes a dual-camera setup with two smartphones placed behind the court along the sidelines to record their videos. Using the cameras of the smartphones is a convenient approach which gives high-quality imagery for detection.

Broadcast TV

Another way to obtain a dataset is to utilize the broadcast videos of the sport. This approach makes it easy to obtain the game videos and also makes it more relevant since the analysis is done on the same video that is observed by a viewer. But, this approach suffers from the varying camera angles and the pan, tilt, zoom (PTZ) motion of the camera in a broadcast video.

The systems in Archana and Geetha (2015), Yu et al. (2004), Pingali et al. (1998), Teachabarikiti et al. (2010), Ekinci and Gokmen (2008) use the broadcast television videos for the detection of a tennis ball.

In Huang and Huang (2017), the broadcast videos are used for detection in the game of badminton.

2.1.2 Camera Calibration

The camera calibration process consists of extracting the extrinsic parameters such as position and orientation in the real world, and intrinsic parameters such as focal length, distortion coefficient and image centre, for the model-estimation of a camera (Kamble et al., 2017).

This is done in Polceanu et al. (2018) and Fazio et al. (2018) by recording videos of a checkerboard pattern (Figure 2.2). In Fazio et al. (2018), a Matlab calibration toolbox is then used to deduce the camera parameters. The authors in Owens et al. (2003) utilize model-based tracking for camera calibration which determines the camera parameters.

2.1.3 Court Mapping

This stage involves field modelling and the extraction of court lines which is useful for some of the detection processes and also for building a line-calling system like the Hawk-Eye system in Owens et al. (2003).

In Owens et al. (2003), local mean removal is performed along with adaptive thresholding to detect line segments. These methods ensure that only straight lines of the court are detected and not the outlines of the players. In Polceanu et al. (2018) and Zhou et al. (2015), colour-based segmentation is used which is followed by a Hough transform for detecting the white lines in a tennis court. This method first detects the white-colored objects in the frame and in those objects finds the straight lines of the court.

Figure 2.2: Camera Calibration using a checkerboard pattern Fazio et al. (2018)

In Fazio et al. (2018), using HSV (Hue-Saturation-Value) thresholding and morphological operations the court lines are detected for tennis. Similarly, in Ekinci and Gokmen (2008) a HSV image is used for court line detection by applying a top-hat morphological operator to the V (Illumination) channel followed by thresholding and erosion. By using the HSV image instead of normal colored image, the operations are much faster as they operate on only one component of the image but these approaches suffer in case of illumination changes or in the case of shadows.

2.1.4 Foreground Extraction

Each frame in a sport video can be divided into two parts: a foreground and a background. The foreground includes the dynamic part of the frame with moving objects such as the players and the ball. These are the set of pixels whose intensity varies during a video. Whereas, the background part mostly remains the same throughout the video and includes the court and court lines, background audience, scoreboard etc. These background pixels change their intensity values minimally during the course of a sport-video. It is therefore necessary to separate the foreground from the background to detect the moving ball in a video frame and that can be done using background modelling or frame-differencing.

Background Modelling

In background modelling, a video sequence of an empty court (without the players or ball) is used to generate a background model that can be further used for differentiating between the foreground and the background. In Mao et al. (2007), the authors discuss a simple background model which is formed by taking the average of the video frames, a method also used by Archana and Geetha (2015). They then improve this model by making it a Gaussian model where each pixel is associated with a mean and deviation parameter. Another approach specified by them is the Mixture of Gaussian (MOG) model where each pixel in the image is modelled to one of the Gaussian distributions that form the image such as the foreground or the background Gaussian distributions. The MOG approach gives more accurate results but is also more time-consuming as compared to the Gaussian modelling or averaging.

In Hrabalík (2017), the background model is created using the median of every pixel. The system in Ekinci and Gokmen (2008) generates the background by applying a median filter over a time duration which results in a static image. Although this technique is efficient, it is computationally-intensive due to the non-linear nature of the median filter.

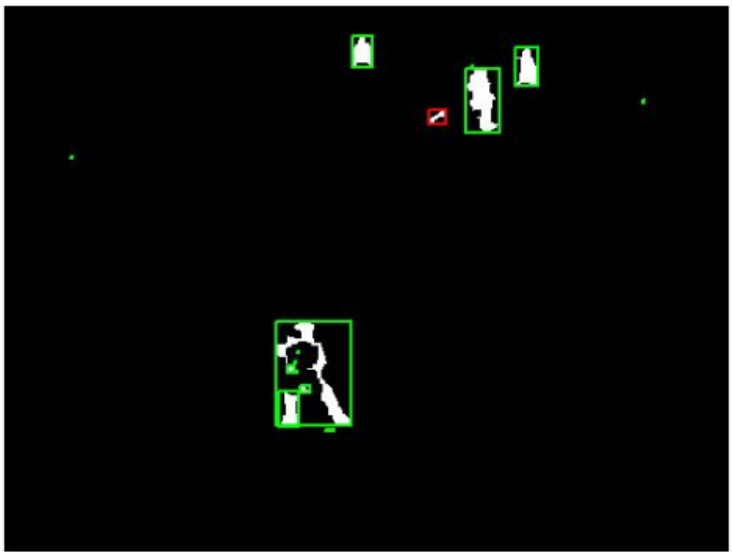


Figure 2.3: Frame Differencing (and detection) performed in Teachabarikiti et al. (2010) where the binary image consists of the foreground objects (in white)– the players and the ball bounded by the differently coloured boxes (detection)

Frame Differencing

Another popular method for foreground extraction involves performing frame differencing between two or more frames to separate moving objects from static ones. This is a popular method as frame differencing takes much less time compared to background modelling and in turn, generates useful results. The authors in Rozumnyi et al. (2017), use three differential images of three consecutive frames to generate one single image which involves the object trajectory. Using three images ensures that only movements that persist from the first to the third frame are recorded.

The systems in Conaire et al. (2009), Teachabarikiti et al. (2010), Pingali et al. (1998) and Pingali et al. (2000) perform consecutive frame differencing followed by thresholding to detect moving components. This is much faster as compared to processing three image-frames but can only be applied to situations where there are no sudden movement in the image-frames.

2.2 BALL DETECTION

Following the separation of the foreground from the background, the next step which comes in the Ball-Tracking problem is to detect the ball candidate in an optimal and accurate manner in an image frame. To do that, various cues from the characteristics of the ball and the game itself can be utilized to make the detection process easier. The features that are exploited commonly in sports for the detection process are discussed in the following sub-sections.

2.2.1 Ball Color

The colour of a ball is distinct to a game. For example, the colour of the ball in the game of tennis has a distinct yellow colour which remains applicable to the game played either on an amateur level or professionally. So, a common way to detect the ball is to perform a colour-based segmentation on the image frame.

Figure 2.4: Size-based detection in badminton by Huang and Huang (2017) where the larger-sized blobs (left) are the players and the smaller-sized blob (right) is the badminton shuttle

This colour-based segmentation approach has been employed in the game of tennis by Yu et al. (2004) where they use a ball colour sieve to filter out candidates that did not have the ball colour pixels.

Pingali et al. (2000) search for the areas in the image which match the intensity value of the ball. In their work, Qazi et al. (2015) look for objects in a yellow coloured plane. This means, a yellow-coloured object (the tennis ball) emerges as a white coloured object in the yellow colour plane and can be detected with ease. All of these approaches use the same idea to detect the ball, only the implementation differs.

In Fazio et al. (2018), Pingali et al. (1998) and Ekinci and Gokmen (2008), they convert the RGB (Red-Green-Blue) color image to the HSV (Hue-Saturation-Value) color space and exploit the HSV value to filter out ball candidates.

Although convenient, the approach to detect a ball by its color suffers from the occasional problems of varying illumination, motion blur, lighting and the player or the background being of a similar color (Kamble et al., 2017). The ball also generally appears as a semi-transparent streak which can affect this process (Rozumnyi et al., 2017). The game has to be recorded using a high-speed and high-resolution camera for the ball to not appear as a transparent-streak.

2.2.2 Ball Size

The size of an object is an important feature for detection. The size of the ball can be computed in pixels (Mao et al., 2007) and can be analyzed using the size (area) of the bounding box around the object, as employed by Ekinci and Gokmen (2008) and Huang and Huang (2017) (Figure 2.4).

The size of the ball can also be pre-computed using a training set as was done by Teachabarikiti et al. (2010) and Yu et al. (2004) and then it can be used as an image mask for detection.

In Archana and Geetha (2015), they first discriminated the largest sized blob in the image as the player blob to detect the ball candidate in the image frame. Pingali et al. (1998) and Owens et al. (2003) also utilize a similar approach to detect a ball candidate.

2.2.3 Ball Shape

The ball shape is widely used as an essential feature for the detection of a ball (Kamble et al., 2017). The shape of a ball can be characterized by a number of different parameters and properties such as:

- Compactness and roughness: The compactness of an object is given by the formula:

$$\text{Compactness} = \frac{4\pi A}{P^2} \tag{2.1}$$

where, A and P represent the Area and Perimeter of the object respectively.

Since, a perfectly circular object has a compactness value of 1, in Mao et al. (2007) objects with a compactness value closer to 0.9 are considered to be the ball candidates.

- Aspect ratio, as used by Archana and Geetha (2015) and Mao et al. (2007). It is given by the formula:

$$A_R = \frac{d_{\min}}{d_{\max}} \tag{2.2}$$

where, d_{min} and d_{max} represent the length and breadth of the object. The aspect ratio lies in the range from 0 to 1 (1 in the case of a perfect circle).

- The height-to-width ratio of the bounding box, as utilized by Yu et al. (2004) and Ekinci and Gokmen (2008).
- Harris Cornerness, which is calculated by correlating the derivatives of an image in the x and y directions. It is used by Fazio et al. (2018), but they don't provide an exact explanation of how the value is utilized for detection.
- Eccentricity, which should be closer to 1 for a circle as used by Qazi et al. (2015).

Even though it is a useful method to filter out the ball candidates from noise, the shape of the ball can vary depending on the frame rate (ball captured as a streak at a lower frame-rate), camera resolution (ball appears semi-transparent at lower resolution), background (noise appears as a ball candidate) and the speed of the ball (ball appears as a streak when hit and appears circular when it slows down) (Kamble et al., 2017). In such cases where the shape of the ball changes in frames due to these properties, applying shape parameters becomes difficult to discriminate between various objects in an image. For them to be utilized correctly, the conditions of the image-frame have to be known before-hand.

2.2.4 Ball Location

The location of the ball candidate can give further cues for better detection. In Yu et al. (2004), the ball is classified in either the upper or the lower region of the frame by using the distinct sound of the hitting of the tennis ball.

The court lines in a game can be detected (or pre-calculated and given as an input) and candidates outside the court lines can then be filtered out as was utilized by Teachabarikiti et al. (2010).

In Zhou et al. (2015), they discard the candidates that are in close-proximity to the players. Such candidates are determined as false candidates by them as they are thought to be noise due to the players movement. In Fazio et al. (2018), they use rays extended through the centroid of the ball to their cameras to evaluate the location of the ball on the court (Figure 2.5).

2.2.5 Ball speed

For games like tennis and squash, the ball moves at a really high speed which can be an important cue when differentiating between noise and ball candidates. This is used in Conaire et al. (2009) for the game of tennis where they employ speed constraints in their process as the speed of the ball must be consistent across image frames.

2.2.6 Ball Motion

Candidates can be filtered out if they do not follow the appropriate motion-model of a ball. In Rozumnyi et al. (2017), they test the candidates detected based on the

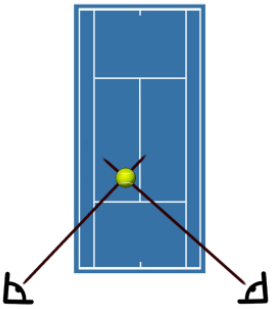


Figure 2.5: Ball Location using extended rays in Fazio et al. (2018)

motion consistent to that of a "fast-moving object." In Zhou et al. (2015), the authors test and evaluate a candidate based on the predicted motion model of the ball. In Yan et al. (2005), they use seed-triplets (candidates of previous, present and next frames) to confirm the motion and detect the ball candidates.

2.3 BALL TRACKING

A ball-tracking algorithm is used to build a coherent motion of the ball across multiple frames which can be represented as a smooth and continuous trajectory of that ball. It also helps in filtering out incorrect ball-detections, makes localization better and handles the challenge of occlusion. There are various methodologies used in the literature to build a tracking algorithm which are discussed in the following sub-sections.

2.3.1 Kalman Filtering

The Kalman Filter (KF) works on linear systems by taking the previous state and the noise as input and estimates the current optimal state of the ball. Although good for linear systems with Gaussian distributed states, the KF performs poorly when there is occlusion (due to the lack of input values) or non-linear motion (due to the linear quadratic estimation of the filter) (Kamble et al., 2017).

For their work on tracking a soccer ball, Kim and Kim (2009) used a Dynamic Kalman Filter (DKF) to tackle occlusion in the system. For every instance of KF estimation, they modified the search area, covariance matrices and targets to be tracked depending on the situation. So, in case of occlusion the target of their KF becomes the player instead of the ball and the search area changes to the area near the player (Figure 2.6). Their DKF performed considerably better than the usual KF.

The KF approach is also used in tandem with a trajectory-based tracking approach as discussed in Section 2.3.3. In Yu et al. (2004) and Chakraborty and Meher (2012b), a KF based verification technique is used for the trajectory-based approaches.

In Fazio et al. (2018), the state of the ball which includes the position and the velocity of the ball is estimated using a Kalman Filter with unknown correspondence. They use a "simple linear physics system as the state prediction model."

The authors in Ekinci and Gokmen (2008), utilize a KF based approach with a trajectory-based approach for the game of tennis. They utilize two 1-D Kalman Filters instead of a single 2-D KF for tracking on the x and y-axis. By using two 1-D KF's, they solve the problem of the sudden change of direction of the ball by the hitting players without keeping track of the player's position. However, they do not specify whether the accuracy of the system remains the same when two 1-D KF's are used in the place of a two-dimensional KF.

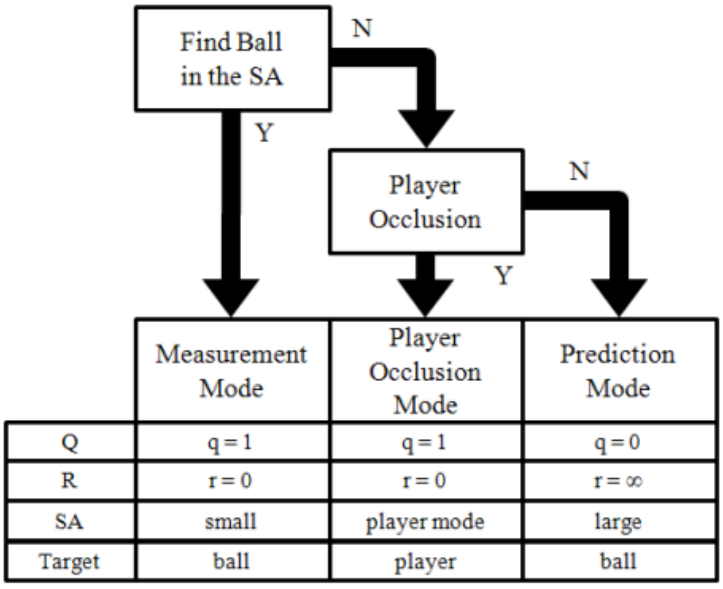


Figure 2.6: Dynamic Kalman Filter in Kim and Kim (2009)

2.3.2 Particle Filtering

Particle Filtering is a suitable approach for systems which are prone to occlusion and have objects with varying sizes and non-constant velocity (Kamble et al., 2017). This approach has mainly been utilized in the game of soccer which has constant occlusion during the game or in the game of table-tennis where there is non-linear motion.

In Huang et al. (2008), a particle-filter based object tracker is used to track small objects. Their particle filter uses motion estimation and a mixture model for handling uncertainties due to occlusion, motion-blur and clattered background.

In the game of tennis, Yan et al. (2005) utilized two separate dynamic models for tracking the tennis ball. This makes the system computationally efficient as only the particles that are in the respective modes are chosen (Kamble et al., 2017).

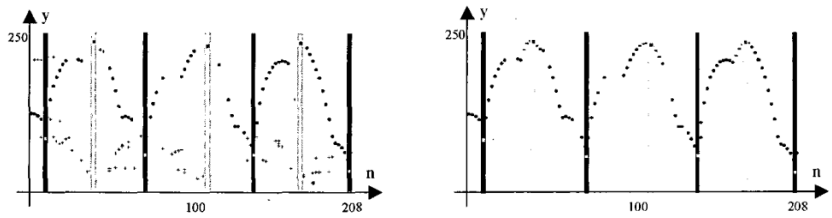
2.3.3 Trajectory based approaches

In trajectory-based algorithms, various techniques that use the generation of a three-dimensional or a two-dimensional trajectory are included. In these techniques, the focus is on determining whether a trajectory is a correct ball trajectory rather than focusing on the optimum detection of the ball in each frame. The main reasoning is that when the ball detection phase returns a lot of false positives, it is easier to check the true positives by fitting them in a ball trajectory rather than filtering the detections in each frame.

In the case of a 3D trajectory, it can be done using triangulation as done in Owens et al. (2003) where they triangulated the three-dimensional position of the ball using multiple high-speed cameras across the tennis stadium. The high-speed cameras detect the position of the ball in two-dimensions in an optimum manner. These 2D detections are then combined at a central processing station in a three-dimensional view of the trajectory.

The authors in Pingali et al. (2000) visualize the 3D trajectory of a tennis serve by creating a 2D trajectory and mapping it to a 3D trajectory by utilizing the extrinsic and intrinsic calibration parameters of multiple cameras.

In Yu et al. (2004), they utilize a candidate feature image (CFI) for their 2D trajectory algorithm (Figure 2.7). A CFI is an image in which features of the image-frame



(a) CFI after detections **(b)** Final Trajectory in a CFI

Figure 2.7: Candidate Feature Images in Yu et al. (2004)

are plotted against the frame number in a single image. They particularly plot the centroid of the ball with respect to the frame number in the CFI. They utilize thirteen frames in a single CFI. They evaluate their trajectory using a Kalman-based verification where they predict a candidate with a Kalman filter and evaluate whether any candidate lies near to the position of the predicted candidate and add that candidate to the trajectory.

In the works of Chakraborty and Meher (2012a) and Chakraborty and Meher (2012b) for the game of basketball and volleyball respectively, a trajectory-based algorithm is followed where the ball candidate position is plotted separately for its X and Y coordinates along time. A prediction function and a Kalman function is used to verify the ball positions and to predict the missing candidates respectively. The trajectory is evaluated based on its length, with the longest trajectory selected as the accurate trajectory and the shorter trajectories are rejected.

In Chen and Wang (2007), they not only utilize the length of the trajectory for evaluation but also discriminate between trajectories by measuring the angle and the distance between two successive candidates in the trajectory. The values of the calculated angle and the distance are deemed accurate if they lie in the range computed using a statistical analysis by the authors.

Similarly, in Polceanu et al. (2018), multiple frames are combined into a single trace which is used to build the trajectory of the ball. They combine ten frames in a single image as they correspond to one-third of a second for a video running at 30 frames-per-second.

To generate the ball trajectory, Qazi et al. (2015) take the distance of the ball from the top left corner of the image and plot this distance with respect to the frame sequences to generate a smooth trajectory.

2.3.4 Data Association

Data-association methods consist of generating a ball-track from uncertain measurements (Kamble et al., 2017). These algorithms work well when tracking fewer objects in a noise-free environment but the process becomes difficult and computationally complex as the number of objects and the noise increases.

For the game of tennis, Yan et al. (2005) use the data association approach to divide the tracking task into 3 layers. The first layer works on the candidate level where the position of the ball candidates is determined using seed-triplets and a sliding-window-based mechanism. In the second layer, the algorithm operates on the 'tracklet' level, where every node in a graph is considered as a 'tracklet' and the graph is solved using Dijkstra's shortest path algorithm. In the last layer, the optimal path is found which has the maximum number of nodes and the least weight.

A similar approach is used in Zhou et al. (2015) where they divide their process into three parts, candidate extraction, 'trajectorylet' generation and global 'trajectorylet' splicing. In the second part, they use a motion model to form the 'trajectorylets' and in the third part, they use Floyd's algorithm to join the adjacent 'tracjetorylets' to form a ball-trajectory (Figure 2.8).

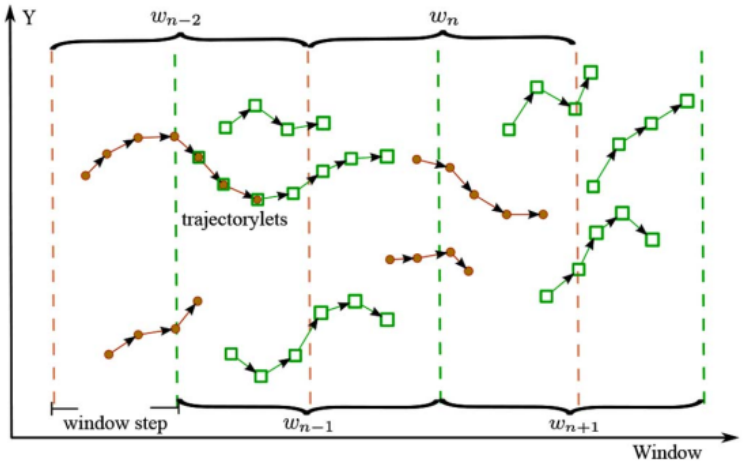


Figure 2.8: Data Association based approach in Zhou et al. (2015)

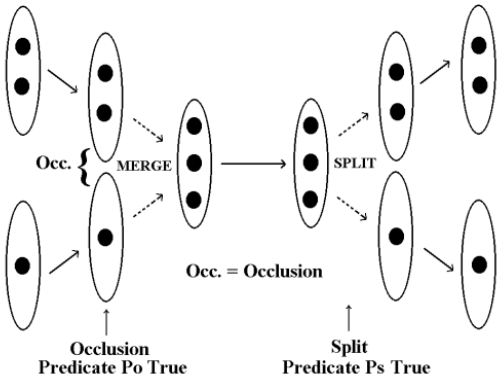


Figure 2.9: Merge-Split approach to handle occlusion (Gabriel et al., 2003)

2.4 OCCLUSION

The problem of occlusion is a common problem in ball-tracking scenarios, where the ball is occluded either by the players or it blends in the background. According to Gabriel et al. (2003), occlusion happens when a blob of one or more objects combines with another such blob to form a single blob which contains both the objects.

In Gabriel et al. (2003), they discuss the two common ways to solve such a problem, the first is a merge-split approach where the objects are continuously tracked until they are occluded and form a bigger collective blob. This collective blob of objects is then tracked until the atomic objects split from it (Figure 2.9). A second approach is a straight-through approach where objects are continuously tracked even when occlusion happens which means every pixel is characterized to an object at every instant of time (Figure 2.10). These approaches use a single camera for tracking.

Out of the tracking algorithms discussed in the previous section, the trajectory-based algorithms are the most suitable in solving the problems of occlusion (Kamble et al., 2017), as they can use motion fitting models for handling the cases of undetected or mis-detected ball candidates. The Kalman filter approach in the duration of long occlusion gives a poor performance with large error rate (Kamble et al., 2017).

Instead of using a single camera, using multiple cameras to capture the different viewing angles can help in solving the occlusion problem. This approach requires

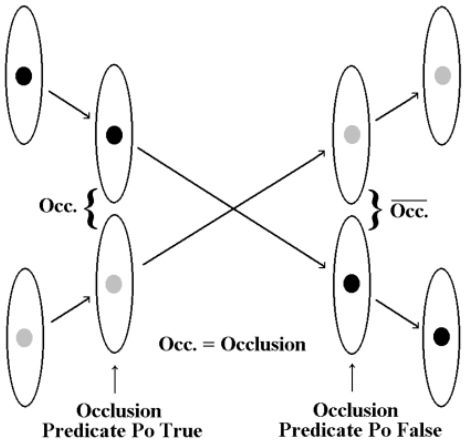


Figure 2.10: Straight-Through approach to handle occlusion (Gabriel et al., 2003)

fusing the data from the different sources and using it to detect the position of the ball. However, this increases the complexity of the system with problems such as synchronization, timing constraints and a need for high-computational resources.

There has not been an adequate discussion of solving the occlusion problem separately in ball-detection systems in the literature. It is assumed, that multi-camera approaches are successful in handling occlusion whereas single-camera approaches tend to not focus on occlusion as it happens infrequently in most of the games.

2.5 CONCLUSION AND HYPOTHESES

Ball-tracking in sports is a difficult problem because of the small size of the ball and the high speed at which it moves. The varying illumination conditions, the shape of the ball and constant occlusion due to the continuous movement of the players makes it even more difficult to track the ball accurately. Various methods present in the literature have been discussed in this section and for every method, it is important to accurately detect a ball in an image frame before continuing onto the tracking part of the process. Various cues related to the ball and the game are used for this purpose such as the color, shape, size, velocity, region and motion of the ball (Section 2.2).

The tracking algorithms (Section 2.3) focus on combining the detections to form a coherent ball-trajectory. These algorithms have evolved over time, starting from the Kalman filter based approaches to trajectory-based methods and the recent mathematical data association methods. For every methodology, there is a trade-off that has to made regarding accuracy, computational complexity and timing. There are solutions that are less computationally complex such as the basic Kalman filter and the trajectory based approaches due to their linear nature. But they suffer slightly in accuracy as they are slow to react to the non-linear situations and occlusion that arises in the system. The methods that focus more on accuracy such as the non-linear filters and data-association methods are computationally intensive due to their excessively mathematical nature. The most successful of the approaches, the Hawk-Eye system Owens et al. (2003), shows that using simple techniques of triangulation can give highly accurate results when combined with a number of high-speed cameras and a high number of computational resources.

For this thesis, which focuses on detection and tracking methods for squash, the processes need to be fast, memory-efficient and accurate. After reviewing the methods present in the literature the following hypothesis are formed related to the objectives of the thesis (Section 1.2):

- A number of previous works utilize pre-processing methods that follow similar steps for extracting the foreground in the image-frame. These include filtering, frame-differencing, binarization and morphological operations. These steps can be utilized for this thesis, if the methods for these steps are chosen considering the low-cost aspects of the project.

- The detection methods present in the literature tend to focus more on the properties of color and shape to detect the ball. But, a squash ball is a small-sized object which moves at very high speeds. This means, it appears as a semi-transparent streak where a color-based segmentation might not be successful. The streak like shape of the ball can be useful to detect the shape but if the shape of the ball changes in different scenarios, then it can not be used as a reliable parameter to detect the ball. Since, a squash ball is the smallest sized object in an image frame, size-based segmentation should provide good results. Region and velocity based-approaches can give good results for detection as the ball remains in the boundaries of the court and its velocity range can be calculated.

- The tracking methods in the literature vary in each scenario. The Kalman filter is the most popular approach since it combines efficiency with accuracy in tracking. Non-linear filters and data-association methods have been used in situations where high-accuracy is required and the system is not limited by the computation memory. For this thesis, since the focus is on memory-efficiency and speed, a Kalman Filter in combination with trajectory-based approaches is a promising approach for obtaining optimal results.

- To compare the results of ball-tracking, a Candidate Feature Image (CFI) as used in Yu et al. (2004), can be utilized to plot the coordinates of the ball against the frame-number. These trajectory plots can then be compared with the detections as in Figure 2.7.

- None of the works reviewed in the literature have been carried out on low-computation platforms such as the Raspberry-Pi, therefore it might not be feasible for the complete process to be carried out under real-time constraints. But, certain optimization steps might be applied to speed-up the process. By modifying the resolution and the frame-rate of the input video, the process could be sped-up to be able to run on the Raspberry-Pi. Although, the effect of such optimization steps on the accuracy of the process needs to be analyzed.

3 PRE PROCESSING

In an image processing or a computer vision application, the pre-processing stage is the initial stage of the process. In this stage, a true-color image (RGB) is taken as an input and the output is an intensity image with the focus on the alteration of the input image to prepare it for further operations. The pre-processing stage includes operations such as enhancement of certain features, removing noise and distortions, and geometric transformations in an image (Sonka et al., 1993).

For this project, the pre-processing stage readies the image frame for the ball-detection and the ball-tracking stages. The main aim in this stage is to extract the foreground of the image frame by subtracting the background in the image. In fact, the two terms, foreground extraction and background subtraction go together as they perform the same task which is the separation of regions of interest in an image frame from the other insignificant details in a frame.

The steps undertaken in this stage can be observed in Figure 3.1

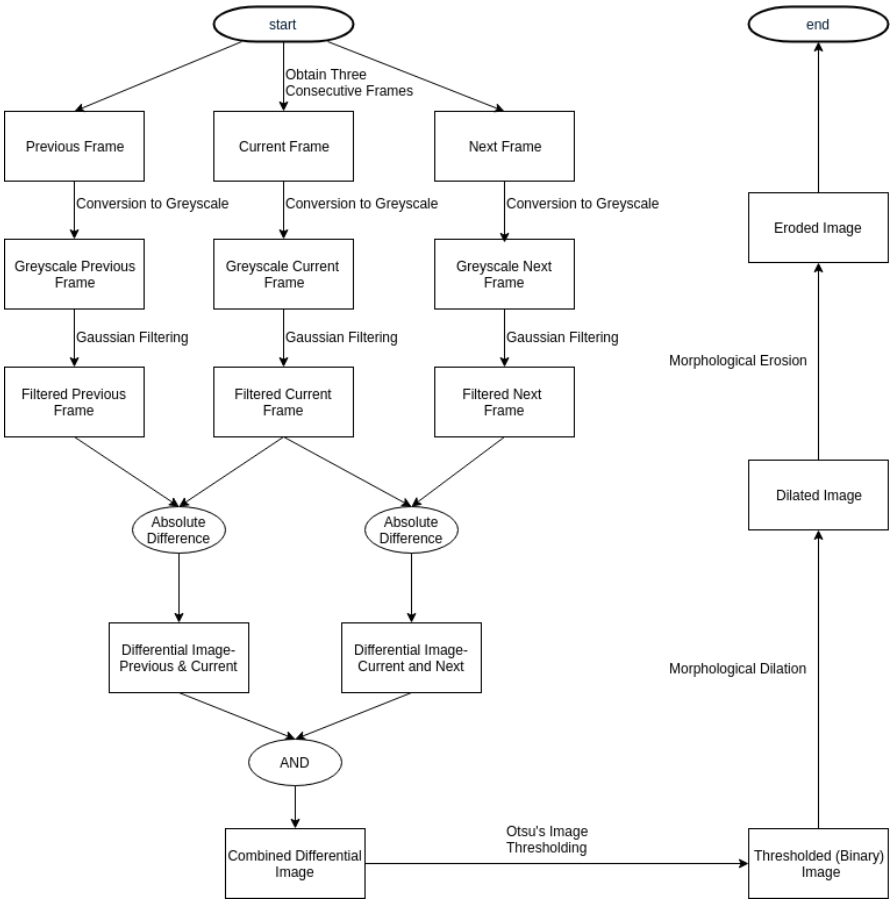


Figure 3.1: Steps undertaken for extracting the foreground

The first step is to convert the coloured RGB (Red-Green-Blue) image frames to their grayscale equivalent to visualize the motion precisely. This is followed by filtering the image frames to reduce noise. The frames are then combined using frame differencing and boolean operations into a grayscale image which captures

the complete motion in an image frame while muting the background. This combined image is thresholded to form a binary image which helps to focus on the objects of interest. The process is concluded by performing certain morphological operations (dilation and erosion) that operate on the shapes in the image by alleviating discontinuity and enhancing the objects in the image.

3.1 DATASET

The standard benchmarks present for object tracking in the field of computer vision do not include fast-moving objects in them (Rozumnyi et al., 2017). To solve this problem, Rozumnyi et al. (2017) created and annotated a new dataset, which consisted of fast-moving objects including data from the game of squash. The squash data in their dataset consists of various angles of horizontal and vertical game shots.

Our problem, which focuses on solving the tracking problem in the game of squash using low-cost equipment, requires the use of a single camera-view for recording the game. This means, the multiple-camera angles in the dataset of Rozumnyi et al. (2017) can not be used for this purpose.

For this single-camera approach, the ideal position to place the camera is behind the squash court at a height. This position is preferred for two reasons:

1. It enables the camera's field of view to cover the entire court region.
2. By placing it behind and above the court area, the camera doesn't interfere in the field of play.

This sort of camera placement to create a dataset can be achieved through a custom camera setup of the camera of choice with a suitable resolution and frame-rate. Another easier way to achieve the same camera-view is by utilizing the broadcast video for squash that is recorded for television. The broadcast TV (BTV) video, has the camera placed behind the court at a certain height which covers the entire court without interfering in the field of play.

Although easy to procure and with an ideal field-of view, the BTV video suffers from the limitations of the choice of the quality of the camera in terms of resolution and frame-rate. Another advantage of a custom-setup is that it can be used to obtain dataset from amateur squash games, which gives a high-variety to the dataset. With BTV video, the only recordings available are those of high-quality professional matches in which the ball and the players move at a very high-speed which makes it difficult to analyze the performance of the algorithm for matches with lower ball-speeds.

A custom camera set-up requires professional aid and expertise which is not possible for such a thesis project. So, to evaluate the algorithm implemented for detection and tracking of the squash ball, the dataset has been generated and annotated for this project utilizing the BTV recordings that are available on Squash TV (2019).

The match recordings that have been obtained are from five different squash matches, that are part of different tournaments. They consist of matches from both male and female players, with squash courts having different colored backgrounds. This ensures that the algorithm is not built in preference to one situation over the other and that it works in the case of different scenarios. The matches that have been used to annotate the datasets are:

- Momen vs Dessouky, CCI International 2019.
- Rodriguez vs Gaultier, JP Morgan Tournament of Champions 2015
- Rachel Arnold vs Nour El Tayeb, PSA World Championship 2018/19

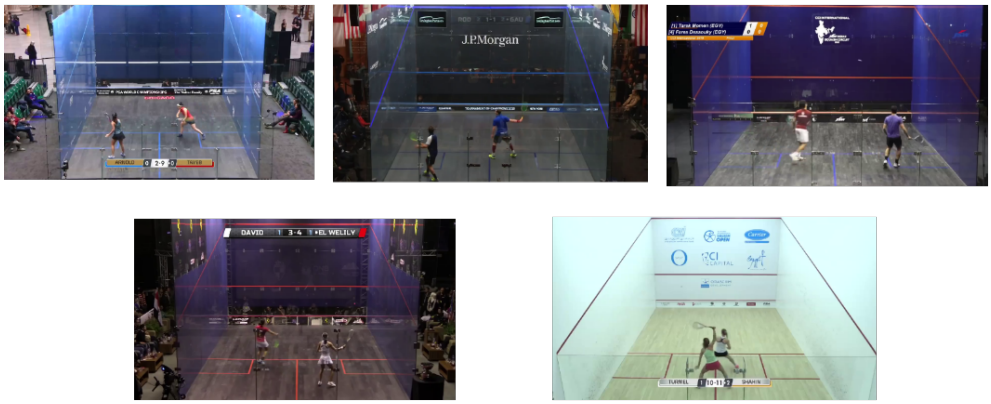


Figure 3.2: Broadcast video matches from which the datasets are extracted

- David v El Welily, Women's World Championship Final 2014
- Lucy Turmel vs Nadine Shahin, El Gouna International 2019

The match recordings are broken into rallies of shots manually. The shot rallies are selected which consist of various player shots from different positions and where the ball suffers from an occlusion in certain frames. Ten of such shot rallies are selected as the datasets from the five different matches. These rallies on an average consist of 250 frames per rally, therefore the dataset consists of around 2500 image-frames in total. The videos run at a frame-rate of 25 frames per second and are of a 720p resolution. However, the resolution of the videos is adjusted during the optimization stage for faster processing as will be discussed in Section 6.2.

3.2 FOREGROUND EXTRACTION

Foreground extraction is the process of separating moving objects in a frame from the static background objects that remain unchanged in every frame. The moving objects are the main focus of tracking algorithms in sports such as tracking a moving player or tracking a high-speed ball or both.

As discussed in the literature (Section 2.1.4), there exists two common ways to perform foreground extraction. One method is to subtract consecutive video frames to identify objects in motion as the pixel values of those objects changes in the corresponding frame implying motion in those pixels. This frame-differencing step is preceded by converting the frames to a grayscale representation and is succeeded by boolean operations to extract the moving parts of the required frame.

Another method is to first build a background model of the scene in the video, either by recording a video for a small duration of time of a still and empty court or by providing the system with a pre-built background model. By subtracting this background model from a video frame, moving objects can be separated in the frame. The frame-differencing approach is considered to be faster whereas the second approach of creating a background model tends to be more robust towards noise.

3.2.1 Conversion to Grayscale

The frame-differencing step is performed after first converting the frames from RGB colour space to a grayscale representation. This is performed using Equation 3.1 as defined by the standard set by the International Telecommunication Union, in their recommendation BT.601 (CCIR 601).

$$Y = 0.299R + 0.587G + 0.114B \tag{3.1}$$

where, R G & B represent the color planes of Red, Green and Blue colors and Y is the resultant grayscale value.

Figure 3.3: Conversion of image frames from RGB to Grayscale

This conversion is done due to two main reasons. One, the luminance component in an image captures more information as compared to the chrominance component and is important to distinguish the various visual features in an image. As in human vision, computer vision looks for features in an image based on contrasting patterns in the image pixels as compared to texture and colour information.

Second, by converting the image frame to a grayscale representation, the complexity of further operations gets reduced significantly. An RGB image is composed of three separate 8-bit channels representing the three colour planes, but a grayscale image is an 8-bit image with every pixel a shade of grey with values ranging from 0 to 255. This conversion simplifies the process and reduces computational requirements (Kanan and Cottrell, 2012).

3.2.2 Gaussian Filtering

Following the conversion to a grayscale representation, the three frames are filtered to remove noise by using a Gaussian filter with a 7x7 kernel size. This blurring step is primarily performed for reducing the image detail and the image noise.

A Gaussian filter is a weighted average filter with a large weight at the centre and smaller weights at the boundary of the kernel. The kernel or the mask represents a smalls-sized matrix which is used to perform convolution over the image. The Gaussian filter is formulated as

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{3.2}$$

where x and y are the distances of the pixel coordinate from the origin (top-left corner of the image) and σ is the standard deviation.

The standard deviation is represented as an inter-pixel space and has an effect on the weights of the elements in the Gaussian kernel. For instance, a large standard deviation corresponds to a greater weight to the boundary elements signifying the effect of the far-off pixels on the average. This leads to a loss of detail in the image along with the generation of noise. A smaller standard deviation, on the other hand, does not have much of an effect as the weights of the pixels off the centre will be small. The standard deviation has been calculated using the kernel size (ksize) with the formula used in the open-source computer vision library by Bradski (2000),

$$\sigma = 0.3 * ((ksize - 1) * 0.5 - 1) + 0.8 \tag{3.3}$$

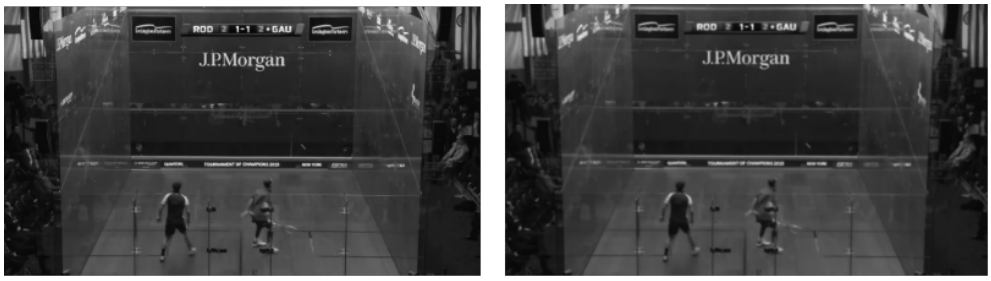


Figure 3.4: On the left- grayscaled version of the image. On the right- result after Gaussian filtering

This means for the kernel-size (ksize) of 7, the σ corresponds to a value of 1.4 pixels.

The size of the kernel pre-dominantly depends on the size of the objects in the image frames. This means, that the kernel size in a filter varies with every image processing/computer vision application. A large sized kernel can be inefficient in removing small salt & pepper noise whereas a very-small sized kernel might filter out the ball candidate from the frame due to the small size of the squash ball. For this application various kernel-sizes were experimented. A kernel-size of value 9 was inefficient in removing noise from the image whereas a kernel-size of value 5 filtered out the ball from the image. Therefore, a kernel-size of 7 strikes the perfect balance to filter out the unnecessary noise in the image frame while maintaining suitable detail in the image to capture the motion.

The implementation of the Gaussian Filter used in this project is of that in the Open Source Computer Vision library (Bradski, 2000). The Gaussian filter coefficients are computed using the formula,

$$G_i = \alpha * e^{-(i-(\texttt{ksize}-1)/2)^2/(2*\sigma)^2} \tag{3.4}$$

where, $i = 0..\texttt{ksize} - 1$ and α is a scale factor chosen such that $\sum_i G_i = 1$. For a $ksize = 7$ and calculating σ from Equation 3.3, the Gaussian filter coefficients are obtained as

$$([0.03125], [0.109375], [0.21875], [0.28125], [0.21875], [0.109375], [0.03125])$$

It can be observed that these are coefficients for a 1-D filter. To obtain the coefficients for a 2-D Gaussian filter for the same dimensions in x and y directions, the coefficient matrix has to be transposed and multiplied to obtain a 7x7 matrix of Gaussian filter coefficients.

An important property used for optimisation is that of separable convolution as the two-dimensional convolution matrix can be separated into two one-dimensional matrices. Since convolution is associative, instead of performing a convolution of the image with the 2D matrix, two separate convolutions can be performed with the 1D matrices in the horizontal and vertical dimensions. The image can be convolved first with the 1D matrix in the horizontal dimension and then can be convolved with the 1D matrix in the vertical direction. Convolution is much faster with single dimension matrices and the results obtained are the same as by a convolution with a 2D matrix.

The choice of a Gaussian filter in this application, rather than an averaging or a median filter is motivated by two reasons – performance and efficiency. A Gaussian filter is a linear filter as compared to the median filter which is a non-linear filter. This means, computation of the new pixel value after convolution with a Gaussian filter is much faster compared to a median filter. Another property of the Gaussian filter is that it doesn't preserve any sharp edges in the image. This is especially useful in the scenario of squash. Squash broadcasts that are used in the project contain a number of advertisement and tournament labelling as shown in Figure 3.5 which when filtered through a median filter introduces noise due to

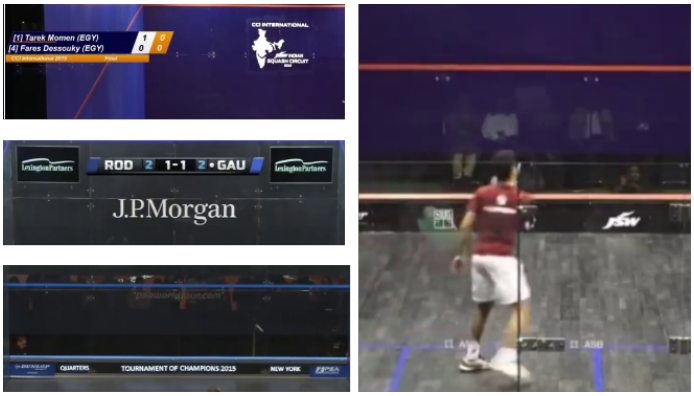


Figure 3.5: The various noise inducing elements in an image frame– Advertisements, Reflections, Labels

the sharp edges of those labels. This noise interferes in the ball-detection stage and makes the detection process more difficult.

3.2.3 Frame Differencing

Frame Differencing is the method of choice to perform foreground extraction in this thesis. Unlike traditional frame-differencing approaches which take the difference of two consecutive frames, the approach in this project rather utilizes two frame-differencing images generated from three consecutive video frames. The reason for such an approach lies in the fact that the game of Squash has many components that can induce noise. A squash broadcast video with all the advertisement and score labels, glass fibre walls with reflections and a transparent front wall with crowd sitting behind them, generates various noisy components that are difficult to differentiate from the significant motion in the frame.

Therefore, for every frame in which motion has to be separated, two other frames, one preceding frame and one succeeding frame are taken in the processing pipeline. Then, two separate frame-difference images are generated by differencing the current frame with the previous one, and by differencing the next frame with the current frame. The results of this process can be observed in Figure 3.6 where the differential images have been inverted to show the results more clearly. The process can be represented in Equation 3.5 and Equation 3.6

$$\delta_{-} = I(t) - I(t-1) \tag{3.5}$$

$$\delta_{+} = I(t+1) - I(t) \tag{3.6}$$

where, $I(t)$ represents an image frame at time t, and δ represents a frame differential image.

The process of frame-differencing is quite fast in performance as it takes very little time in execution which makes it an optimal approach to isolate motion in a frame.

3.2.4 Boolean Combination

Each frame-differential image in Figure 3.6 contains motion from two consecutive frames. These frame-differential images when combined together with a boolean AND operation contains, as a result, motion only from a single frame– the centre of the three frames (Equation 3.7).

$$\delta = \delta_{-} \wedge \delta_{+} \tag{3.7}$$

(a) Frame-difference of 1st and 2nd frames

(b) Frame-difference of 2nd and 3rd frames

Figure 3.6: The two (inverted) frame differential images– δ_- and δ_+

Figure 3.7: Boolean combination of the (inverted) frame differential images– δ

This step ensures the absence of any static objects present throughout those three frames while retaining the motion present in the current frame. This can be observed in Figure 3.7, where unlike the differential images in Figure 3.6, the (inverted) frame-differential image contains only the moving objects present in the frame that is being analyzed. Whereas, the images in Figure 3.6 contains information from both consecutive frames.

3.3 THRESHOLDING

The grayscale image obtained from the previous step has the foreground and the background pixels represented as shades of grey with values ranging from 0 to 255. Here, a pixel value of 0 corresponds to a black color and a value of 255 corresponds to a white color in the image. For further operations, the image needs to be segmented further to separate the foreground from the background. One of the most efficient methods to segment an image is by performing image thresholding.

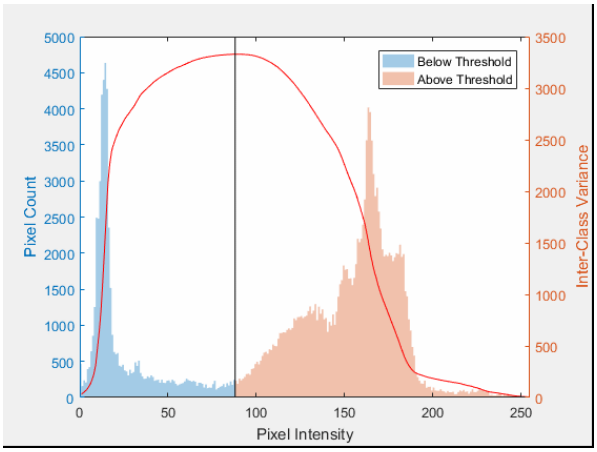


Figure 3.8: Visualzing Otsu's thresholding by minimizing the intra-class variance between classes of pixels

The image thresholding process converts an input image into a binary image. It does that by converting all values above a threshold value to a binary value of one and the values below the threshold to a binary value of zero, separating the pixels in the foreground from those in the background. As in most computer vision applications, the game of squash also contains only a certain number of pixels that are of interest for the purpose of tracking. These are the image pixels representing the squash ball and the players. By thresholding and keeping the value of only these pixels as one, it becomes easier to detect and track these objects.

There are various commonly used methods to perform the image thresholding operation on an image. These methods involve either the user providing the threshold value such as the Simple Thresholding method, or methods which derive the optimum threshold value accordingly, such as the Adaptive Threshold method. The simple thresholding method works for the cases where the knowledge of a global threshold value is known before-hand. The adaptive threshold method is more applicable to cases where the threshold value is expected to change quite frequently and is never constant, such as in images with a lot of varying information (per frame) in it.

Another thresholding method which is used extensively and is ideal for our application is the Otsu's Thresholding method. Otsu's binarization method works well for a bimodal image, where there is a distinct separation between the foreground and background of the image, displayed by the histogram representation of the pixel values in the image. It calculates the optimum threshold value such that it divides the set of pixel values into two separate classes of pixels. This can be visualized in Figure 3.8, where the threshold value divides the pixels into the foreground and background pixels. The threshold value lies in the middle of the two peaks of pixel values in the histogram.

$$\text{Intra-class variance} \quad \sigma_\omega^2 = \omega_b * \sigma_b^2 + \omega_f * \sigma_f^2 \tag{3.8}$$

$$\begin{aligned} \omega_b &= \sum_{i=0}^{t-1} p(i) \\ \omega_f &= \sum_{i=t}^{L-1} p(i) \end{aligned} \tag{3.9}$$

$$\mu_b = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_b}$$
$$\mu_f = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_f} \tag{3.10}$$

The way the method works is by minimising the intra-class variance (as represented in Equation 3.8) of the image for a threshold value. The threshold value is calculated by iterating over the pixel intensity values from 1 till 255 and calculating the weight (ω_b, ω_b), mean (μ_b, μ_f) and the variance (σ_b, σ_f) of the two set of classes (background and foreground) being created by that threshold value. The weight and the mean of the two set of classes are calculated using Equation 3.9 and Equation 3.10, where t is the threshold value and $p(i)$ represents the normalized frequency of the gray-level i. The total number of gray-levels are represented by L.

The threshold value which results in the minimum intra-class variance (Equation 3.8) is chosen as the final threshold value of the image. This method is better represented in Algorithm 3.1.

Algorithm 3.1: Otsu's thresholding

Output : Final Threshold t
Input : Greyscale Image
Initialize: Weight $\omega_i = 0$ Mean $\mu_i = 0$;
1 **for** *t=1 to max intensity value* **do**
2 | Update ω_i *and* μ_i;
3 | Compute $\sigma_w^2(t)$;
4 **end**
5 Desired threshold corresponds to the minimum $\sigma_w^2(t)$;

The Otsu's method is the ideal choice to perform the thresholding operation in the project, as in each frame there is always a majority of pixels with the value near to zero that correspond to the background in the frame, and a very few pixels with larger values (greyish-white pixels) that represent objects with motion in the frame. This can be visualized in the histogram Figure 3.9, where a large number of pixels are congregated near values of zero, and only a really small set of pixels have non-zero finite values. The method, in this case, selects a threshold value of 13 to optimally separate the foreground from the background. After obtaining the threshold value, the pixels in the image are modified to a binary (inverse) value of zero or one accordingly as shown in Equation 3.11.

$$dst(x,y) = \begin{cases} 0 & \text{if src(x,y)} > \text{threshold} \\ 1 & \text{otherwise} \end{cases} \tag{3.11}$$

The final thresholded (inverted) image can be observed in Figure 3.10, where the moving objects (represented in black) in the frame are clearly separated from the non-moving background (represented in white) in the image. The Otsu method is fast as it operates on arrays of fixed length and is quite efficient. The disadvantage of the method is that it assumes uniform illumination in the image, therefore in cases of sudden illumination changes the thresholding method performs poorly and generates noise in the image.

3.4 MORPHOLOGICAL OPERATIONS

Binary images contain multiple imperfections and distortions in terms of object structure caused by the thresholding operation. The image is also infused with

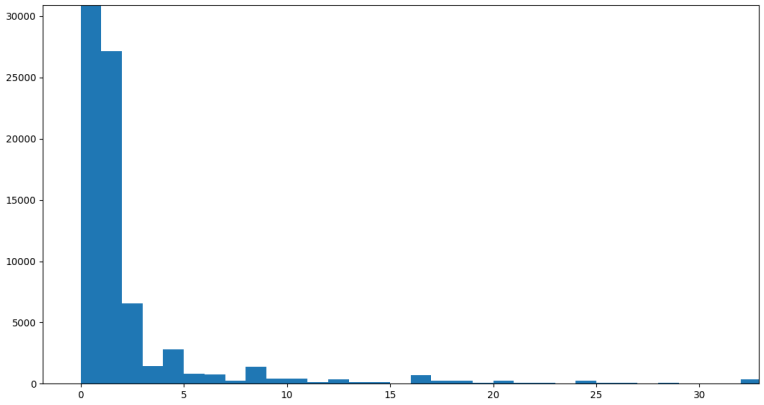


Figure 3.9: Histogram of pixel values in the combined frame-difference image, where the x-axis represents the pixel value and the y-axis represents the number of pixels of that value.

Figure 3.10: Inverted thresholded image using Otsu's binarization

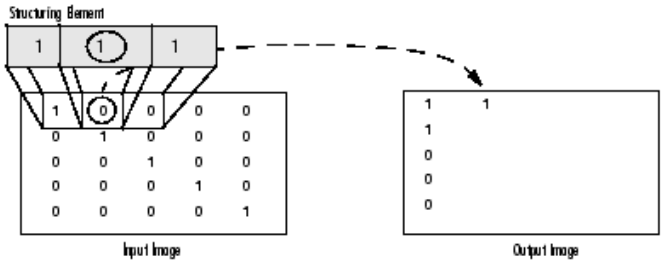


Figure 3.11: Morphological Dilation over a Binary image (Dilation, 2019)

noise, necessitating further operations to be applied on the binary image to refine it further. This is where morphological image processing procedures are utilized.

The morphological transformations operate on the shape and structure of the image. They are non-linear operations that apply to the texture of a binary image for operations such as edge-detection, enhancement, segmentation etc. This is performed with a structuring element (a small 2D array of pixels) which operates over the image and modifies a pixel value accordingly. The most common of these set of operations are – Dilation and Erosion.

The operations of dilation and erosion are commonly performed one after the other, with their order determined by the kind of shapes present in an image. When erosion is performed before dilation, it is termed as morphological "opening" as this opens up gaps in an image when the pixels in an object are connected in a weak manner. The opposite, morphological "closing" is the compound step of performing dilation first followed by erosion. This step connects the objects that are weakly connected while maintaining the initial sizes. For the application of squash, morphological closing is the ideal operation as the thresholded image consists of a lot of disconnected components that need to be connected together first by dilation, and their size modified with erosion afterwards.

The morphological closing operation can be mathematically expressed in Equation 3.12, where S is the structuring element acting on the binary image B resulting in dilated image D, and operating on D to obtain the final image F. $\ominus$ and $\oplus$ symbolize the erosion and dilation processes respectively.

$$F = S \ominus (S \oplus B) \tag{3.12}$$

3.4.1 Dilation

Morphological dilation, as the name suggests expounds the objects in the binary image by adding pixels to the foreground. A structuring element of a small size iterates over the image pixels, assessing the neighbouring pixels of the origin pixel, and sets the pixel values in the structuring element to '1' if one or more pixels in the matrix correspond to a value of '1'. This can be better visualized in Figure 3.11, where the output pixel gets set to 1 because its neighbouring pixel in the structuring element has the value '1.'

Mathematically, morphological dilation can be explained using Equation 3.13 (Dougherty and Lotufo, 2003), where a structuring element S is applied on segments of the binary image B and combined to output the dilated image. $\bigcup$ signifies the union of all the fragments of the image where the element is applied to.

$$S \oplus B = \bigcup_{s \in S} B_s \tag{3.13}$$

The results from dilation on the binary image of Figure 3.10 can be observed in Figure 3.12. Notice how the disconnected structure of the players in Figure 3.10 are now connected as solid and complete objects, making them easier to detect.

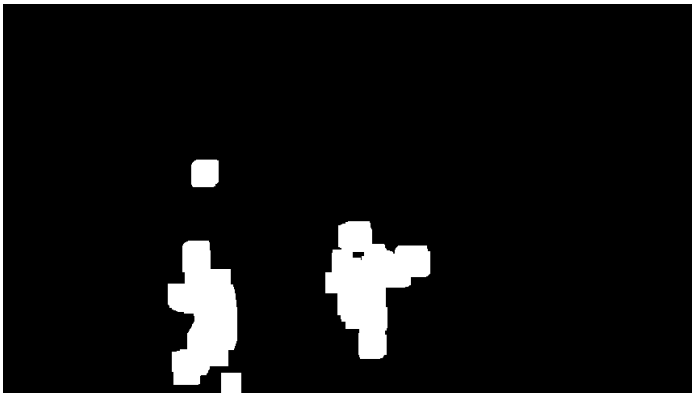


Figure 3.12: Results of Morphological Dilation

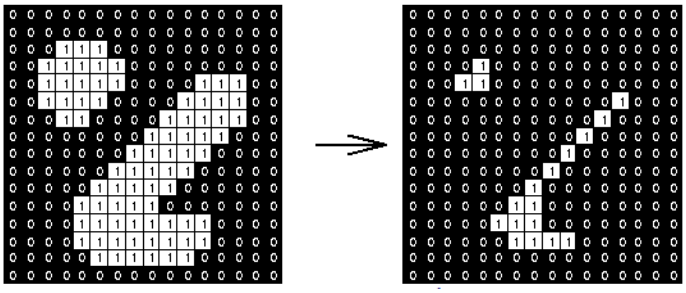


Figure 3.13: Visualization of Morphological Erosion (Erosion, 2019)

3.4.2 Erosion

Morphological erosion is the dual operation of dilation and is mainly used to filter out noise in an image. In it, a small structuring element operates over the image and sets the value of the origin pixel in the structuring element to '0' if any of the neighbouring pixels have the value of '0'. This means noisy pixels with value '1' are converted to '0' since their neighbouring pixels are in the background. The erosion procedure can be observed in Figure 3.13, where a 3x3 structuring element is applied to an image. Erosion can be represented by Equation 3.14, where the structuring element S erodes the segments of the binary image B. $\bigcap$ denotes the intersection of all the segments where the structuring element is applied to.

$$S \ominus B = \bigcap_{s \in S} B_s \tag{3.14}$$

The results of the erosion operation on the dilated image in Figure 3.12, can be observed in Figure 3.14. The primary purpose of applying erosion to the dilated image in this case is to reduce the dimensions of the objects in the dilated image.

3.5 CONCLUSION

The steps taken to separate the foreground objects from the background have been discussed in this section. Since, squash is such a high-paced sport, three consecutive images are taken in the process pipeline for foreground extraction. These images are combined to form frame-differential images and as a result a final binary image is formed. A combination of morphological operations are then performed to "fill-out" the spaces in the binary image.

Each method in this stage has been chosen considering the low-cost aspect of the project. The use of frame-differencing, Gaussian filtering, Otsu's thresholding ensures that the process is fast and memory-efficient. However, considering that

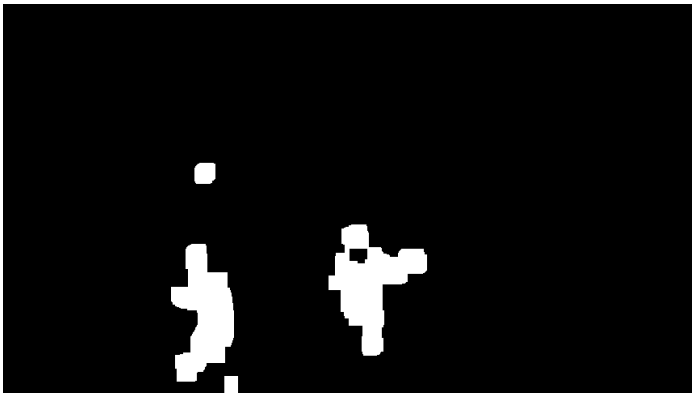


Figure 3.14: Results of Morphological Erosion

the images are large arrays of pixel values, the processing time of this stage is the maximum amongst all the ball-tracking stages, as will be discussed in Chapter 6.

4 BALL DETECTION

The pre-processing step performed on the image frames of the input video, as discussed in Chapter 3, yields a binary image with the foreground of the image separated from the background. The foreground of the image consists of moving objects in each frame. These are mainly the players, the ball and the noise generated by the movement of the players and the ball. The next step in the ball-tracking process after foreground extraction is to detect the ball candidate(s) out of these foreground objects in the image frame.

The best approach to categorize an object as a ball candidate out of the various segmented objects is to utilize the cues from the game and the properties of the ball. Properties such as colour, size, shape, speed, location (Section 2.2) are significant clues to help identify the ball in the image. These properties of a ball are distinct to each ball-game and differs from one game to the other.

In the game of squash, the ball is a small-sized object which moves at high speeds. The colour of a squash ball can vary. It is mainly chosen so as to distinguish the ball from the colour of the court and the player's apparel. The most popular of them is a white-coloured ball used in dark-tinted squash courts by professional athletes. On lighter background courts such as in amateur games, a black ball is commonly used. A squash ball has a diameter of 39.5 to 40.5 mm (Ian McKenzie, 2017) and moves at speeds greater than 200 kmph giving it a streak-like shape while in motion.

The characteristics of the ball used for detection in this project are the size, region and velocity of the ball. Colour and shape of the ball, although popular in the literature (Section 2.2), are inaccurate cues to detect a squash ball. The squash ball used in a professional setting travels at such a high-speed that in a moderate camera setup, it is captured as a semi-transparent streak. This means that a colour based segmentation results in failure to detect the ball. Moreover, since the colour of the ball varies from courts and levels of plays, using colour as a property to distinguish the ball isn't an appropriate approach. The shape of the ball is also an unreliable cue to detect the ball. The squash ball when hit moves at higher speeds which gives it a streak-like shape but when the ball reflects from the walls, it slows down considerably and the shape turns to circular outline. This means a common shape parameter cannot be used throughout the shot to categorize the ball.

The steps involved in the ball-detection process are outlined in Figure 4.1. The first step is to find the contours of the objects in the frame characterized by a similar intensity value. Once the contours are detected, the objects are segmented by the size of the contour which is described by the zeroth image moment. The objects after size based segmentation are divided into three categories – ball candidates, player candidates and incomplete player candidates.

The next step is to eliminate the ball candidates lying outside the court region. This step is followed by filtering out the ball candidates that lie in close proximity to a player as such candidates are generally a flailing part of the player that has been incorrectly segmented as a separate object in the pre-processing stage. The last step in the detection process is to eliminate candidates that violate the velocity constraint of the ball. The motion of the ball is consistent across frames and candidates defying that motion are generally noisy candidates.

The result of applying these detection steps on an image frame are significantly reduced number of ball-candidates. Ideally, only a single ball candidate per frame should be present after this process but there can be images with no candidates at all due to occlusion/misdetection or multiple candidates due to over-segmentation

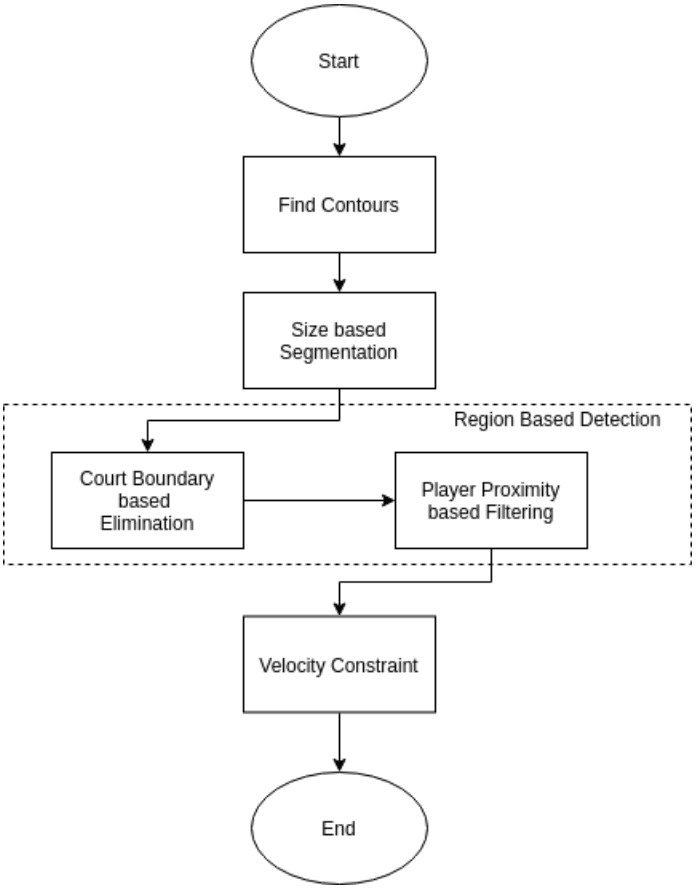


Figure 4.1: Process Overview of Ball Detection

as well. These cases are then managed in the ball-tracking phase in Chapter 5 when detections in multiple frames are combined to form a smooth final trajectory of the ball.

4.1 CONTOURING

A solid object with a smooth surface, when captured in an image, is bounded by an image curve which is called as the *contour* or the outline of that solid object (Forsyth and Ponce, 2003). In image processing terms, it is the curve joining the points having the same intensity values which forms the border of an object. A binary image with image pixel values of 0's and 1's is especially convenient to find the contour. Thus, a common procedure is to first convert an image into a binary image using thresholding (Section 3.3) or Canny edge detection (Harris et al., 1988). Finding the contours in an image is useful in various scenarios such as object detection, topological analysis and image compression (Suzuki et al., 1985).

For a binary image, the 1's signify the objects in the image whose contour needs to be formed and the 0's represent the background of the image. The method to find the contours in an image in this project has been derived from Suzuki et al. (1985). They proposed a border following algorithm for topological structural analysis of an image. Their algorithm uses an effective border labelling method along with a border following technique to effectively map an outer-border joining the 1-pixel components. They do this by keeping track of the parent border of every border and labelling each border uniquely. The method can be visualized in Figure 4.2.

The implementation of the method to find contours by Suzuki et al. (1985) utilized in this project is provided by the open-source computer vision library by Bradski

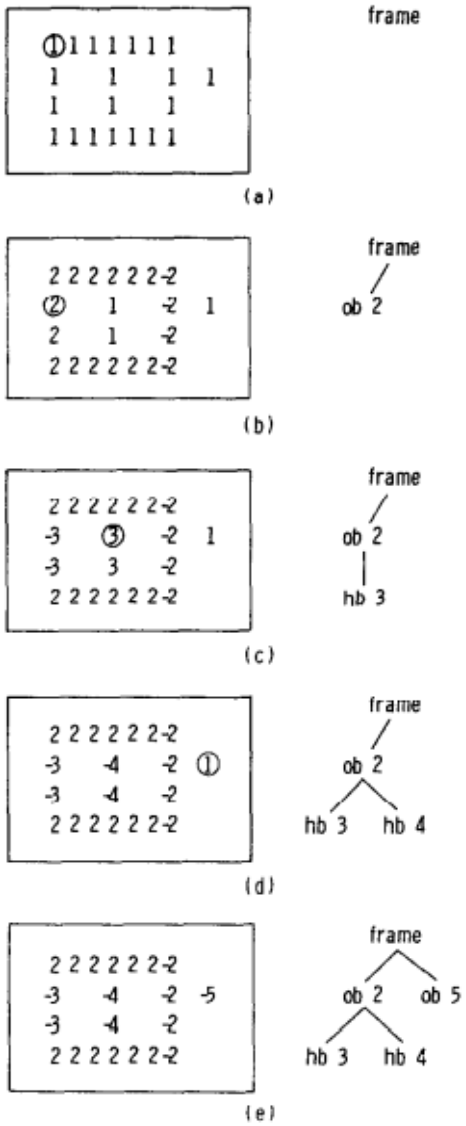


Figure 4.2: Topological Structural Analysis by Suzuki et al. (1985). The circled element represents the start of a new border (outer or inner) in the algorithm.

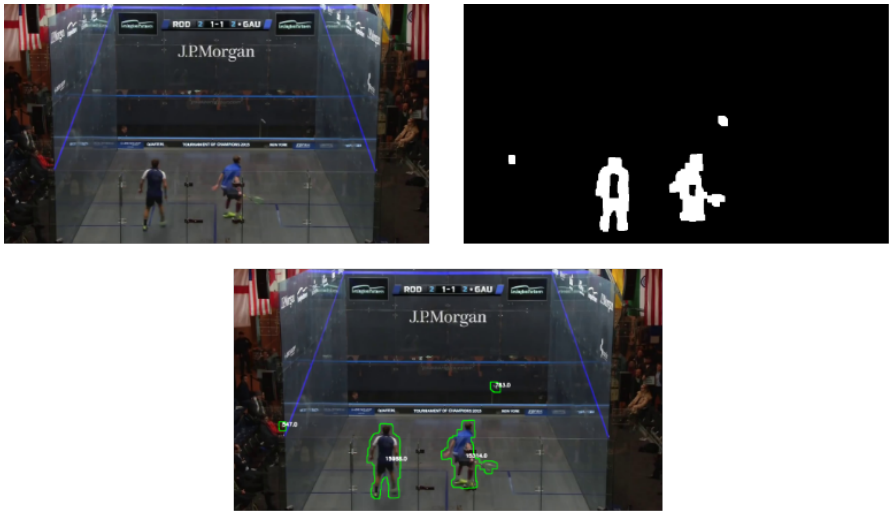


Figure 4.3: Finding and Drawing Contours in a binary image

(2000). It takes the image as an input and returns a matrix of the coordinates of the border points of the contour depending on the input parameters provided. The parameters determine the type of the contour and the number of the points in the contour to be returned by the method. This project requires to map only the outer boundaries of an object and since, not every point in the contour is useful to form the contour, an external contour and a contour approximation parameter are used. Therefore, the method returns only a few coordinates of the external boundary for every object. This step saves a substantial amount of memory in the system and is fast in execution.

The result of the procedure is a list of contours in an image frame. The contours in the list can then be used separately or collectively for other operations such as drawing, finding convexity or calculating image moments. The results of the process of finding and drawing a contour can be visualized in Figure 4.3.

4.2 SIZE-BASED DETECTION

The contours of the objects facilitate the estimation of the size of the objects in the frame by calculating the area enclosed by the contour. The size measurement of the objects can help discriminate between the players and the ball. Size-based filtering is used extensively in the literature (Section 2.2.2) and is a coherent step to determine the ball candidates out of all the objects in the image frame.

Since the moving items in an image frame are the players, the ball and noise, it can be classified that the objects with the largest size in the image will be the players. So, by utilizing a threshold value of the size, the objects can be classified into categories such as *Player Candidates* and *Ball Candidates*.

The threshold value can not be a global value that can be applied to the different squash courts. The threshold value is highly reliant on the location of the camera in the court. The size values of objects in pixels will differ if the camera is placed further behind or forward in the court. However, since the camera for broadcast video footage is placed at a location so that it could optimally cover the entire court as well as maintain premium viewing angles, the threshold value comes out to be similar for most of the BTV videos.

When using a custom camera setup, the thresholds are calculated in the calibration stage using the camera parameters. But since the dataset used in this thesis is from the BTV videos, the most optimum way to determine the size threshold is to perform a statistical analysis of the dataset. By calculating parameters such as the

mean, deviation, minimum and maximum of the size of the objects in a dataset, an indication of the range of sizes in a BTV video can be obtained. The results from the statistical analysis by measuring the size of the objects in Dataset 1 and Dataset 2 can be observed in Table 4.1 and Table 4.2. The size values in the analysis are in pixels.

Table 4.1: Statistical Analysis of Size values (in pixels) of Dataset 1

Candidates	Mean	Std Dev	Min	Max
Ball Candidates	868	292	307	2035
Incomplete Player Candidates	7726	2869	2355	12090
Player Candidates	19650	2828	10018	26266
Player Candidates (single blob)	40039	4460	31736	48499

Table 4.2: Statistical Analysis of Size values (in pixels) of Dataset 2

Candidates	Mean	Std Dev	Min	Max
Ball Candidates	739	239	340	1383
Incomplete Player Candidates	4609	1979	1845	9358
Player Candidates	14262	2271	9289	17407
Player Candidates (single blob)	26396	3572	16524	32722

The statistical analysis in Table 4.1 and Table 4.2 shows that the size values lie in a specific range for the different objects in the frame. It can also be observed that the optimum threshold values differ from one dataset to the other due to the position of cameras in the datasets.

The objects in each dataset are classified into three categories – player candidates (the largest objects), ball-candidates (the smallest sized objects) and incomplete player objects. These incomplete player objects are players in the image frame that are over-segmented during the pre-processing stage and therefore have a size range in between the size of the players and the ball candidates. There are also cases when only a single large-sized object is detected in the frame instead of two separate player objects. This happens when the two players happen to be in immediate vicinity of each other during a shot. In such a case, the pre-processing step is not able to segment the players separately and the result is one large blob which contains both the players together. Since, the size of these blobs is greater than the size of a single player, they are considered as player candidates as well in future steps.

The size of the contours is calculated using the Image Moment. The image moment is a quantitative measure of the shape of the image. It is described mathematically as,

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y) \tag{4.1}$$

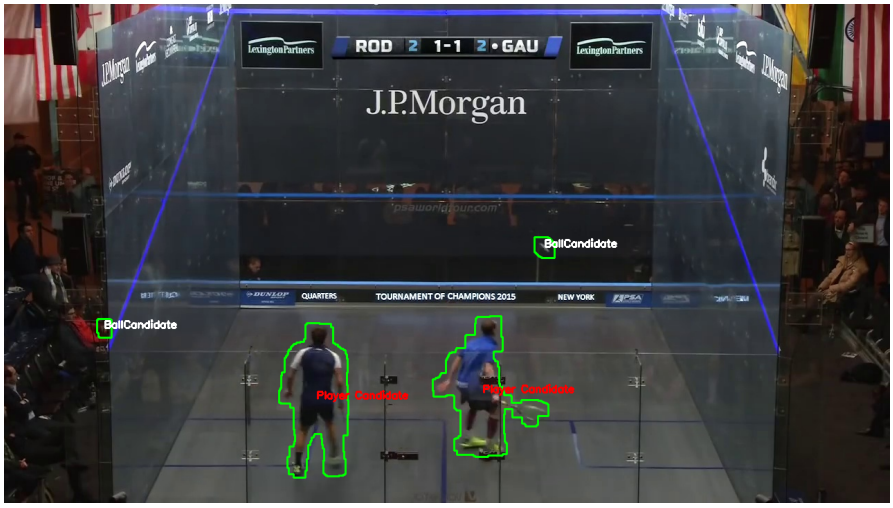


Figure 4.4: Segregating objects by their sizes

where, $I(x,y)$ is the image represented by its pixel intensities and $i+j$ is the order of the moment. To calculate the area, the zeroth moment is required as shown in Equation 4.2

$$\begin{aligned} M_{00} &= \sum_x \sum_y x^0 y^0 I(x,y) \\ M_{00} &= \sum_x \sum_y I(x,y) \end{aligned} \tag{4.2}$$

For a binary image, $I(x,y)$ is '1' for an object in the image thus effectively adding to the area of the object.

The threshold values from the statistical analysis are initialized according to the dataset from which they are derived from. These values are compared with the area of each object to classify the objects into either a player, ball or an incomplete player as shown in Algorithm A.1. The objects with size less than the minimum size of the ball are discarded as infinitesimal noise.

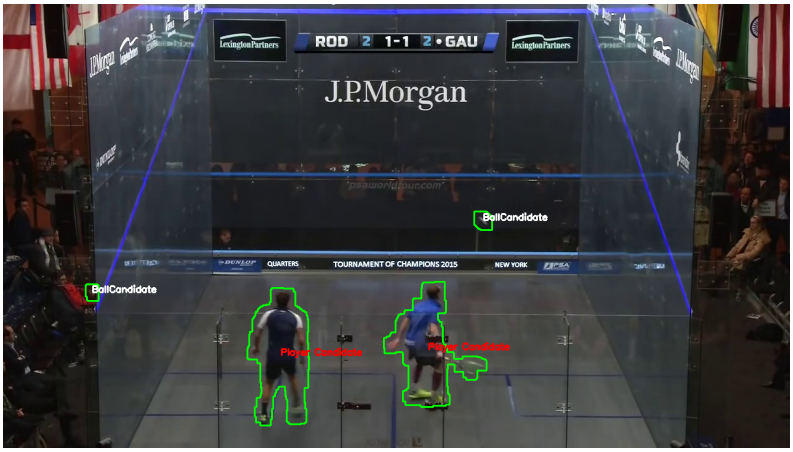
The result of the process is each object segregated according to their sizes in separate containers, that can be used for further processing.

4.3 REGION-BASED DETECTION

The location of the object in an image frame is an important cue to classify it as a legitimate candidate for detection. An object should always be within the bounds of the playing region and always be at a certain distance from the player, to appropriately classify it as a true ball candidate. This means, certain measures can be put into place to filter out the candidates that do not satisfy the above-mentioned characteristics. The following sub-sections discuss these candidate elimination criteria in detail.

4.3.1 Court-Boundary Based Elimination

The position of the cameras placed to record the broadcast videos in squash is not regularized. This means, that they can be placed a little further behind the squash court and can capture the spectators sitting on the sidelines. Any small and fast movement by the crowd gets captured during the pre-processing stage and passes the size-based detection phase as well. This is shown in Figure 4.4 where the movement in the crowd is captured as a ball candidate.



(a) After size-based segmentation, a ball candidate is detected outside the court boundary

(b) Court-based segmentation removes the candidate outside the court boundary

Figure 4.5: Eliminating candidates using the squash court boundaries

The coordinates for the boundaries of a squash court are fixed values that are predetermined either through a calibration process or can be provided to the system by analyzing the dataset. The algorithm to filter out the candidates in this step is illustrated in Algorithm A.2.

The result of this procedure can be observed in Figure 4.5.

4.3.2 Player proximity based filtering

The approach to classify an object as a legitimate ball candidate using the location of the candidate can be further utilized by filtering out candidates that lie in extreme proximity of the players. Such candidates are not the true ball-candidates but in reality, are a flailing part of the player's body that has been segmented incorrectly as a ball candidate.

This happens when the sudden movement of the hand or the feet of the player is captured as a separate moving object that the player. This "over-segmentation" of the players results in candidates that match the size description of a ball candidate and induce noise as false positives in an image frame.

Such an approach has been utilized in the literature as well for tennis where in Zhou et al. (2015) the detection close to the players were filtered out. A statistical analysis on the distance of the ball to the closest player candidate in a dataset confirms that commonly a ball never gets closer than a minimum distance to the player.

This can be observed in Table 4.3 for datasets 1 and 2, where the ball is always at some distance to the player. Albeit, there can arise few situations when the ball does get extremely close to a players body, but those situations have been infrequent as observed from all the datasets.

Table 4.3: Statistical Analysis of distance of the ball to the closest player

Dataset	Mean	Std Dev	Min	Max
Dataset 1	224.59	53.47	89.36	342.23
Dataset 2	262.96	88.60	127.32	489.58

The algorithm to filter the candidates based on their distance from the players is described in Algorithm A.3. In an image frame, there can be cases when the players are not segmented thoroughly as the two players in the frame. There can be over-segmented players represented as incomplete players and under-segmented players represented as a single player object. The algorithm takes into account all of these three cases when calculating the distance from the players and the minimum distance is used as the distance to the closest player.

The distances are measured from the centroids of the objects. The coordinates of the centroid of an object in terms of image moments are represented as $\{\bar{x},\ \bar{y}\} = \left\{\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right\}$, where

$$\begin{aligned} M_{10} &= \sum_x \sum_y x I(x,y) \\ M_{01} &= \sum_x \sum_y y I(x,y) \\ M_{00} &= \sum_x \sum_y I(x,y) \end{aligned} \tag{4.3}$$

M_{10} and M_{01} are the summation of x and y coordinates of an object respectively. These when divided by the total number of pixels (the area - M_{00}) gives the average value which is the centroid of the object.

The results of this procedure can be observed in Figure 4.6, where the hand and the shoulder of the player which was classified as a ball candidate in previous steps are now filtered out and only the true ball candidate remains.

4.4 VELOCITY CONSTRAINT

The squash ball follows a certain motion where the distance it moves (in pixels) from one frame to the other lies in a certain range of values. This distance per frame translates to the velocity of the ball candidate. By putting a constraint on the velocity of a ball candidate, several noisy candidates that appear unexpectedly in the image frame or those that don't follow the motion of the ball can be filtered out.

The statistical analysis on the motion of the ball on some datasets in Table 4.4 shows that the ball moves a distance more than a minimum distance in every frame

Table 4.4: Statistical Analysis of the velocity of the ball

Dataset	Mean	Std Dev	Min	Max
Dataset 1	20.23	13.64	2.23	98.11
Dataset 2	24.99	19.70	2.82	105.72

(a) Minimum Distance of ball-candidates with the players

(b) Removing candidates that lie in close proximity to the player

Figure 4.6: Eliminating candidates based on extreme proximity to a player

and never exceeds a maximum value of distance travelled. The maximum value comes when a player hits the ball to make a shot. These minimum and maximum thresholds can be utilized to filter out noise candidates.

In an image frame there can be four cases that can occur:

1. There is no ball candidate detected in the image.
2. There is a single correct ball candidate detected in the image.
3. There are candidate(s) detected neither of which are the ball candidate.
4. There are multiple candidates detected which contain the true ball candidate as well as noise.

When correlating a ball candidate in the current frame to the candidate in the previous frame, several combinations of the above cases can occur. The only case where the procedure fails is when a correct ball candidate is detected in the current frame, but the candidate(s) in the previous frame are incorrect ball candidates. This leads to a False Negative in the current frame which is a drawback of this procedure. It is expected that the ball-tracking stage which correlates the detections across all of the frames can deal with such cases while forming the final trajectory of the ball.

The algorithm to carry out this procedure is described in Algorithm A.4. For cases when there are no detections found in the previous frame, all the detections in the current frame are retained so that the true ball candidate doesn't get lost. The results of this procedure can be observed in Figure 4.7, where the candidate in the lower half of the image frame which is caused by a reflection of the player's movement, is correctly classified as a false ball candidate since it does not move according to the range of motion of the ball during the frames, whereas, the true ball candidate's motion is within the range of the thresholds and is classified correctly.

Figure 4.7: Using a velocity constraint to classify true ball candidates from the false ones

4.5 RESULTS

The results of the steps taken to perform the ball detection procedure are characterized by the two principal parameters – Precision and Recall. The model evaluation metrics of precision and recall capture the efficiency of the detection algorithm much better compared to the parameter of accuracy in this case. The chief underlying problem with the accuracy parameter here is that the number of ball candidates is heavily outnumbered by objects that are not the ball candidates. This means even if every candidate is classified as a non-ball candidate, the accuracy of the system will be very high but that won't paint the true picture of the fundamental application of the algorithm. Thus, the main objective here is to locate the True Positives amongst the detections.

The true positive in the dataset is the candidate that is labelled as a ball-candidate and is actually the ball in the image frame. The other two values important to calculate precision and recall are the false cases – false positives and the false negatives. The false positives are the candidates classified as a ball-candidate whereas that candidate is in fact not a ball-candidate. Similarly, the case of false-negative is when a ball is not identified as a ball-candidate by the algorithm whereas it is the true ball-candidate in the image. The metrics are defined by the formulas in Equation 4.4 and Equation 4.5.

$$\text{Precision} = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4.4}$$

$$\text{Recall} = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.5}$$

The precision metric signifies the relevant results amongst the total percentage of results whereas recall implies the capability to classify the total relevant results by the algorithm. So, a high recall means that the algorithm is able to classify a high number of relevant candidates in the dataset whereas a high precision value signifies that a high number of candidates classified as relevant by the algorithm are actually relevant (Shruti Saxena, 2018).

The results of the detection algorithm at every stage of the detection process are shown in Table 4.5.

The thresholds that have been calculated from the datasets 1 and 2 in Section 4.2 - Section 4.4 are used for the remaining datasets. These two datasets represent as

"training datasets" for the rest of the datasets. This is possible due to the placement of the camera in each dataset in such a way so as to optimize the viewing experience of the broadcast. This means, that the cameras in every dataset are placed at similar positions. But, since the camera position in each dataset is not exactly the same, the threshold values are modified by a few pixels to optimize the results.

The precision value can be observed increasing after every detection step, whereas a high recall value is observed in every step of the detection process. The high-recall value signifies that the algorithm is able to classify a high amount of relevant candidates in each dataset. This is an advantageous attribute of the system as this signifies that the algorithm doesn't miss out on a candidate that could be a ball candidate. The increasing precision value signifies that with each step in the detection process the ability of the algorithm to classify candidates as the true ball candidates keeps increasing, resulting in a more *precise* classification with each step.

An improved evaluation metric to classify the accuracy of the system in binary-classification problems is the **F-measure** or the **F1-score**. The F1-score is the harmonic mean of the precision and the recall values as shown in Equation 4.6.

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.6}$$

This means that it takes into account both the precision and recall to output a number between 0 and 1 representing the accuracy of the system. A value of one symbolizes an optimal balance of precision and recall values and a value of zero implies the opposite.

The final F1-scores of the datasets are separately shown in Table 4.6.

The F1-measures for the datasets are observed to be on an average close to 85%, implying that the detection algorithm has a high accuracy and is able to classify the ball-candidate correctly in most of the cases. These results hold even more significance when taken into account that they are obtained without using the properties of colour and shape of the ball in the game which are the primary properties used in the literature for sports such as tennis. There can be two substantial observations that can be taken away by the detection process:

1. Ball Detection in the sport of Squash which has high-occlusion can give positive results if correct properties regarding the game and the ball are used.

2. A high-accuracy in the detection algorithm can be achieved even without using the notable detection methods of colour-based segmentation and shape analysis present in the literature.

4.6 CONCLUSION

The methods that are used to detect the ball-candidate from the foreground of the image have been discussed in this section. It was observed that the popular methods in the literature which use the property of color and shape to detect the ball were ineffective in squash. Instead, size, region and the velocity of the ball were used for detection purpose. This demonstrates that for varied applications, using a common approach doesn't solve the specific challenges in those applications.

The results after applying the detection methods have been evaluated using the F1-score and a high value of F1-scores were observed for almost all datasets. The only dataset for which the value of the F1-score declined was for a squash match played on a white coloured court using a black ball. This was due to the fact that when the ball moves it becomes a semi-transparent streak, and detecting this streak over a white background becomes quite challenging. In fact, this is one of the reasons that most of the professional squash matches are played on a dark-tinted glass court with a white ball.

Table 4.5: Detection Results for datasets – Step by step

Dataset	Detection Step	Precision(%)	Recall(%)	F1 Score(%)
Dataset 1	Contouring	26.82	96.24	41.95
	Detection by size	50.87	95.02	66.26
	Detection by region	56.78	95.92	71.33
	Detection by velocity	78.16	86.37	**82.06**
Dataset 2	Contouring	22.62	96.7	36.66
	Detection by size	42.31	99.44	59.36
	Detection by region	56.36	99.44	71.94
	Detection by velocity	84.54	94.08	**89.06**
Dataset 3	Contouring	17.46	88.89	29.19
	Detection by size	46.67	88.89	61.21
	Detection by region	72.56	88.89	79.9
	Detection by velocity	73.07	88.37	**83.52**
Dataset 4	Contouring	29.19	89.03	43.97
	Detection by size	69.46	90.09	78.44
	Detection by region	77.4	91.02	83.66
	Detection by velocity	91.11	87.16	**89.09**
Dataset 5	Contouring	22.28	76.87	34.55
	Detection by size	51.11	77.77	61.68
	Detection by region	57.06	70.94	63.25
	Detection by velocity	68.75	66.89	**67.81**
Dataset 6	Contouring	21.63	89.83	34.86
	Detection by size	47.05	86.85	61.03
	Detection by region	66.08	86.36	74.87
	Detection by velocity	84.39	82.95	**83.66**
Dataset 7	Contouring	17.19	94.78	29.1
	Detection by size	36.36	91.86	52.1
	Detection by region	54.7	91.86	68.57
	Detection by velocity	85.36	84.54	**84.95**
Dataset 8	Contouring	27.34	94.8	42.44
	Detection by size	68.47	93.88	79.19
	Detection by region	75.08	92.54	82.9
	Detection by velocity	88.2	89.77	**88.98**
Dataset 9	Contouring	24.46	93.63	38.79
	Detection by size	53	88.99	66.43
	Detection by region	61	89.4	72.52
	Detection by velocity	83.33	85.64	**84.47**
Dataset 10	Contouring	24.84	95.81	39.45
	Detection by size	56.93	91.46	70.18
	Detection by region	68.08	90.99	77.89
	Detection by velocity	87.5	89.57	**88.52**

Table 4.6: Detection Results- F1 Score

Dataset	Number of Frames	F1-score (%)
Dataset 1	560	82.06
Dataset 2	215	89.06
Dataset 3	140	83.52
Dataset 4	330	89.09
Dataset 5	210	67.81
Dataset 6	180	83.66
Dataset 7	220	84.95
Dataset 8	240	88.98
Dataset 9	245	84.47
Dataset 10	235	88.52
	Total Frames= 2575	Average F1-score= 84.21

The performance of the detection process can be improved by using a custom-camera setup with a higher frame-rate. With a setup of lower frame-rate, the ball is often observed as a semi-transparent streak and detecting this blurred-streak over a glass wall or a white background in a squash court is quite challenging. The datasets in this thesis have a frame-rate of 25 frames-per-second, but a camera with 60 FPS can capture the ball much more accurately which can result in better detection results.

5 BALL TRACKING

The ball tracking process combines the detections in each frame from Chapter 4 to form a final coherent trajectory. It keeps track of every detection, starting from the first frame and predicts the position of the ball when there are no detections or helps in determining the actual ball when there are multiple candidates in a frame. This means, at the end of the process there emerges only a single ball candidate in each frame. These candidates can then be combined to form the final trajectory.

In this thesis, ball tracking has been performed using two different approaches. The first approach is motivated by the literature study and uses a Kalman Filter for tracking purpose. The second approach is a novel approach in tracking which is one of the prime contributions in this thesis. In it, a form of exponential smoothing, named Holt's Double Exponential Smoothing (Holt, 2004), is used which takes in account the trend of a trajectory to perform better tracking. The approach to perform ball tracking is same in both the cases, it is only the prediction method which changes in each case.

The complete tracking process can be observed in Figure 5.1. The first step is to determine the initial position of the ball. There can be three scenarios for the first image frame: 1) There is no ball candidate detected, 2) There is a single ball candidate detected and 3) There are multiple ball candidates detected. For every case, the initial position is determined using distinct ways:

1. No Candidate: Use the center of the image frame as the initial state.
2. Single Candidate: Use the candidate detected as the initial state.
3. Multiple Candidates: Use the candidate closest to the center of the image frame as the initial state.

The center of the image is taken as the reference for the initial states. This is because in the 2D space of the image frame, when the shot rally starts, the players serve the ball standing in their service box which in a 2D space lies adjacent to the center of the image.

For the rest of the shot sequence, the scenarios that arise are same as in the first image frame, 1) No ball candidate 2) Single ball candidate 3) Multiple ball candidates. For each scenario, the tracking process is carried out in the following manner:

1. No Candidate: The ball location is predicted using a Kalman Filter or the exponential smoothing technique.
2. Single Candidate: The location of the candidate is measured and assumed to be the real ball location. The Kalman filter prediction is corrected based on the measurement. Similarly in the case of exponential smoothing, the measurement information is used for updating the estimates.
3. Multiple Candidates: The ball candidate which is detected close to the prediction having its distance to the predicted position less than a certain threshold is chosen for measurement. This measurement information is then used for correction/estimation.

A drawback of this whole process lies in the fact that when a single candidate is detected, it is assumed to be the true ball candidate. There are (rare) cases when that

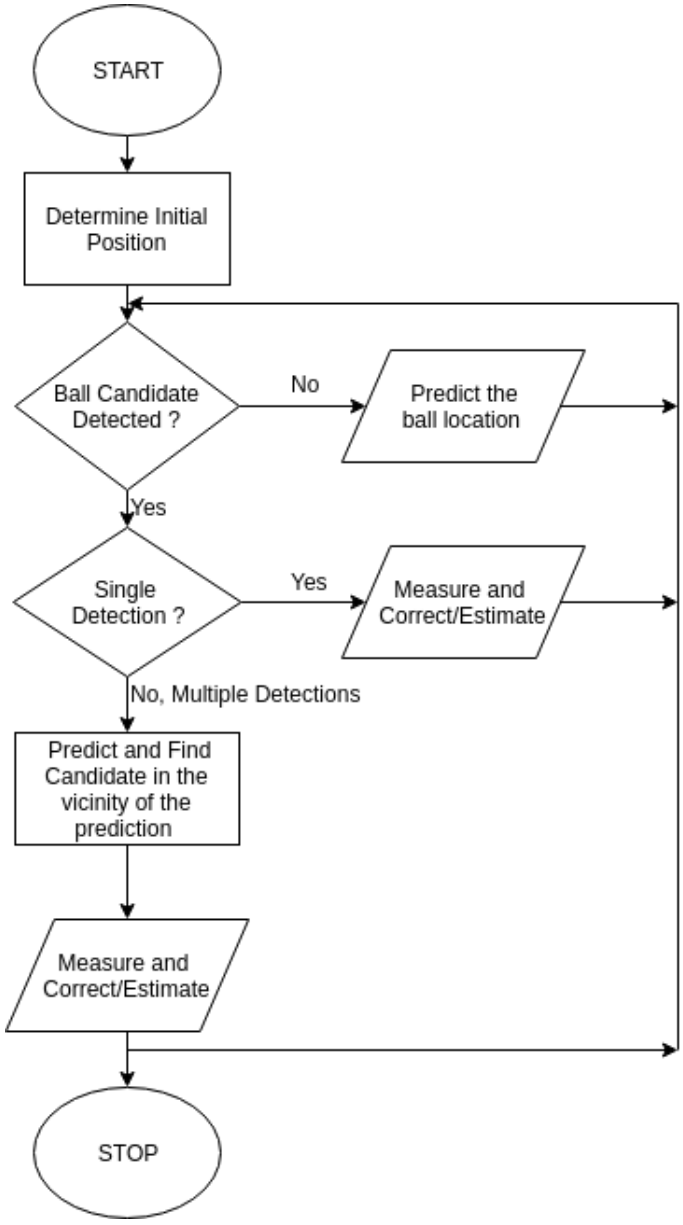


Figure 5.1: Ball Tracking Process

is not true, and the single candidate that is detected in the image frame is a form of noise. However, this assumption is the basic foundation of the whole process and is necessary for carrying out the tracking process. Therefore, it cannot be eliminated from the process.

In the following subsections, the functioning of the prediction methods is discussed and their results are shown.

5.1 USING A KALMAN FILTER

A Kalman Filter is one of the most popular estimation methods in various fields of technology. It is a linear filter which takes into account the previous estimate, current measurement and noise to output the optimal current estimation of a state. This can be understood using Equation 5.1 (Welch et al., 1995), where the predicted state $\hat{X}_k$, depends on the measurement Z_k, previous estimate $\hat{X}_{k-1}$ and the Kalman Gain K_k. Here, the difference $(Z_k - \hat{X}_{k-1})$ is termed as the "residual" as it shows the difference between the actual measurement and the prediction.

$$\hat{X}_k = \hat{X}_{k-1} + K_k(Z_k - \hat{X}_{k-1}) \tag{5.1}$$

A KF is especially helpful in the case of ball-tracking in squash. For most part, the squash ball follows a linear trajectory and has a consistent motion that can be used to predict the position of the ball in case none or multiple detections occur. Also, the implementation a Kalman Filter is considerably light on computation-memory as compared to a non-linear filter (Section 2.3.2) or the more-mathematical data association approaches (Section 2.3.4), which makes it suitable for this thesis.

The drawbacks of using a Kalman Filter is its inability to optimally model the non-linear situations that arise during tracking. When there is a sudden change in the direction of motion of the ball after a player/wall hit, a KF can take time to update the parameters. Also, when the ball is not detected for a large number of consecutive frames, the uncertainty in measurement information reduces the performance of the filter tremendously. Altogether, there exists a trade-off in using a KF for this application. The KF performs especially well in most parts but suffers in quality during long sequences of occlusion or during complicated motion of the ball in shot rallies. Since, such situations arise significantly fewer times in a shot rally and due to the ease of implementation, a KF based approach is the most utilized approach in ball-tracking (Section 2.3.1).

There are two important steps that precedes the process of a KF. The first is to define a state variable which is being estimated in the process. In this case, the state variable will consist of the position and the velocity of the squash ball. In a 2D space, it consists of the position p and velocity v in both x and y directions, as shown in Equation 5.2.

$$X_k = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{5.2}$$

The second step is to define the motion model that is used in the filtering process. In this case, the kinematic equations with a constant velocity are utilized for the motion model. This is represented in Equation 5.3. Here, p and v represent the position and velocity of the ball, and ϵ represents the noise. The value of Δt for consecutive image frames is 1.

$$\begin{aligned} p_{x_t} &= p_{x_{t-1}} + v_{x_t}.\Delta t + \epsilon \\ p_{y_t} &= p_{y_{t-1}} + v_{y_t}.\Delta t + \epsilon \\ v_{x_t} &= v_{x_{t-1}} + \epsilon \\ v_{y_t} &= v_{y_{t-1}} + \epsilon \end{aligned} \tag{5.3}$$

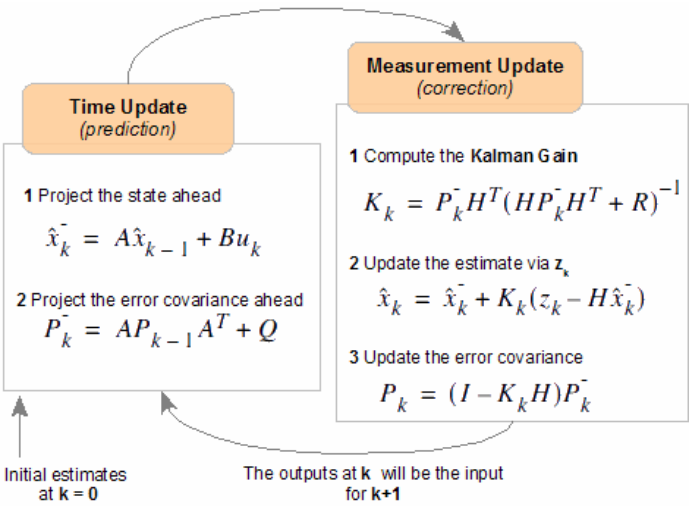


Figure 5.2: Prediction and Correction stages in the Kalman Filter (Esme, 2009)

The working of the Kalman Filter can be represented in Equation 5.4.

$$\begin{aligned} x_k &= F_k x_{k-1} + B_k u_k + w_k \\ z_k &= H_k x_k + v_k \end{aligned} \tag{5.4}$$

Here, F_k is the state-transition matrix which can be derived for the state x_k using the motion model in Equation 5.3 and comes out to be

$$F_k = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

u_k is the control vector and since there is no external control signal in the scenario, it is set to be zero. H_k is the measurement matrix which consists of the state that can be observed in the model and is used for the measurement z_k. In this case, only the position of the ball i.e. its x and y coordinates can be measured. Therefore,

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.6}$$

w_k and v_k represent the process noise and the measurement noise in the system represented by their covariance matrices Q_k and R_k respectively. Q_k is the process noise covariance matrix which represents the noise in the whole process and can be tuned to balance the accuracy and the lag in the process. In this process it is taken as,

$$Q_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * 0.09 \tag{5.7}$$

Since, the information that can be measured in the image frame is certain to be accurate, the value of measurement noise covariance matrix is considered to be very low

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * 0.00003 \tag{5.8}$$

The Kalman Filter works in two stages. In the first stage, it predicts the state ahead and projects the error covariance. In the second stage, it computes the Kalman Gain using the predicted error covariance, measures the actual state and

updates the estimate and the error covariance. This whole process can be observed in Figure 5.2.

As discussed at the starting of the chapter, the measurement information for correction and estimation of the parameters is obtained in the cases of single/multiple detections. When there is no ball candidate detected in a frame, only the prediction step is performed. This means, for a long sequence of poor/no ball detections, the error covariance matrix doesn't get updated which leads to inadequate predictions.

5.2 USING HOLT'S DOUBLE EXPONENTIAL SMOOTHING

Exponential smoothing is a form of moving-average method where the weights assigned for calculating the average decrease gradually from the most recent observations to the past observations. It was introduced in Brown (1957) where using Equation 5.9, the smoothed value s_t can be calculated using the observation x_t, the previous smoothed value s_{t-1} and the smoothing parameter α.

$$s_t = \alpha \cdot x_t + (1 - \alpha) \cdot s_{t-1} \tag{5.9}$$

Brown's exponential smoothing works well for stationary time-series where the smoothed value lies around a constant mean but not for series' which have a trend. Therefore, the smoothing method was extended in the work by Holt (2004) where the trend of the data was taken into account as well. This meant, both short-term and long-term movement of the data in a particular direction can be incorporated to smoothen and predict the next value in a time-series.

In Holt's double-exponential smoothing method, smoothing is performed twice, once for calculating the estimate for the level in the series and the second time to calculate the trend in the series. This is represented in Equation 5.10,

$$\begin{aligned} L_t &= \alpha x_t + (1 - \alpha)(L_{t-1} + T_{t-1}) \\ T_t &= \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1} \end{aligned} \tag{5.10}$$

The level L_t at time t is calculated using the observation x_t, smoothing parameter α and taking in account the previous level and trend values, L_{t-1} and T_{t-1} respectively. The trend T_t is calculated using the levels at time t and $t-1$, smoothing constant β and the trend at time $t-1$. The forecast $\hat{X}_{t+m}$ for mth value beyond time t is made using the level and trend values at time t, as shown in Equation 5.11.

$$\hat{X}_{t+m} = L_t + mT_t, \tag{5.11}$$

Both level and trend smoothing equations have separate smoothing constants, α and β. α signifies the weight assigned to the recently observed values, which means a high value of α corresponds to more weight given to the recent observations and less weight assigned to the previous observations. Since, in squash things can change rapidly during a shot-sequence, a high value of α is chosen for this thesis. The significance of the value of β is analogous to that of α. A high value of β signifies shorter trends in the data. Since, in squash the trend continues longer as the ball doesn't changes its trajectory at every moment, the value of β is chosen to be small in this thesis. The exact values have been derived using an experimental approach where the values of α and β were set to the maximum and minimum respectively, and were altered accordingly to obtain optimal results.

In the ball tracking process, the level and trend estimates are updated only when detections occur. In case of no ball detection, only prediction takes place using the last level and estimate values.

Table 5.1: Ball Tracking Results- Number of frames where the ball was predicted vs detected

Dataset	Number of frames	Frames with ball detected (%)	Frames with ball predicted (%)
Dataset 1	560	79.47	20.53
Dataset 2	215	87.45	12.55
Dataset 3	140	70	30
Dataset 4	330	74.85	25.15
Dataset 5	210	28.58	71.42
Dataset 6	180	75	25
Dataset 7	220	77.73	22.27
Dataset 8	240	76.67	23.33
Dataset 9	245	73.47	26.53
Dataset 10	230	74.49	25.21

5.3 RESULTS AND COMPARISON

In the 2D trajectory generated as a result of the tracking process, the y-coordinates of the centroid of the ball are plotted against the image frame number. This is a form of a Candidate Feature Image as introduced in Yu et al. (2004).

A quantitative score to classify the results of the methods cannot be effectively used since it is really challenging to create the ground-truth to compare the results. The ball moves as a blur at such high speeds, that in most of the cases it is very difficult to locate the ball in the image-frame from an unaided perspective. This means, to compare the trajectories generated from the two methods, a qualitative approach has to be utilized to observe the results.

The qualitative characteristics that classify a ball -trajectory as a preferred trajectory are as followed:

- The points in the 2D plot should lie inside the image boundaries. This means, that in a 480p-resolution image frame, the y-coordinate of the ball should lie in the range of 0-480.
- The trajectory should be smooth. The position of the ball changes gradually in a shot and the presence of peaks in the trajectory shows lack of ball-detections which implies uncertain behaviour.

The 2D trajectories for Dataset 1 and 2 can be observed in Figure 5.3 and Figure 5.4. The trajectory results for other datasets can be found in Appendix B. Moreover, the results of the tracking stage on each dataset can be observed in Table 5.1, where the number of frames where the ball had to be predicted in each dataset (in %) is recorded as compared to the frames where the ball location was detected.

The observations that can be made from the results above:

1. There are occasions when the y-coordinate values in the plot using a KF approach in both the datasets lie outside the image boundaries. This doesn't happen when using the smoothing technique as it takes into account the trend of the data and quickly adjusts itself when the trend of the data changes.
2. The trajectories in the Holt's exponential method are also smoother compared to the Kalman method. This is expected as it is a form of "smoothing" technique after all, where more weight is given to the recent observations.
3. The final trajectory for Dataset2 is better as compared to Dataset1 as Dataset2 has better detection results compared to Dataset1. This signifies that better detection results contribute greatly to the final tracking results. This fact can also be observed by associating the results in Table 5.1 with the trajectories in Figure 5.3, Figure 5.4 and Appendix B. It can be seen that datasets having a

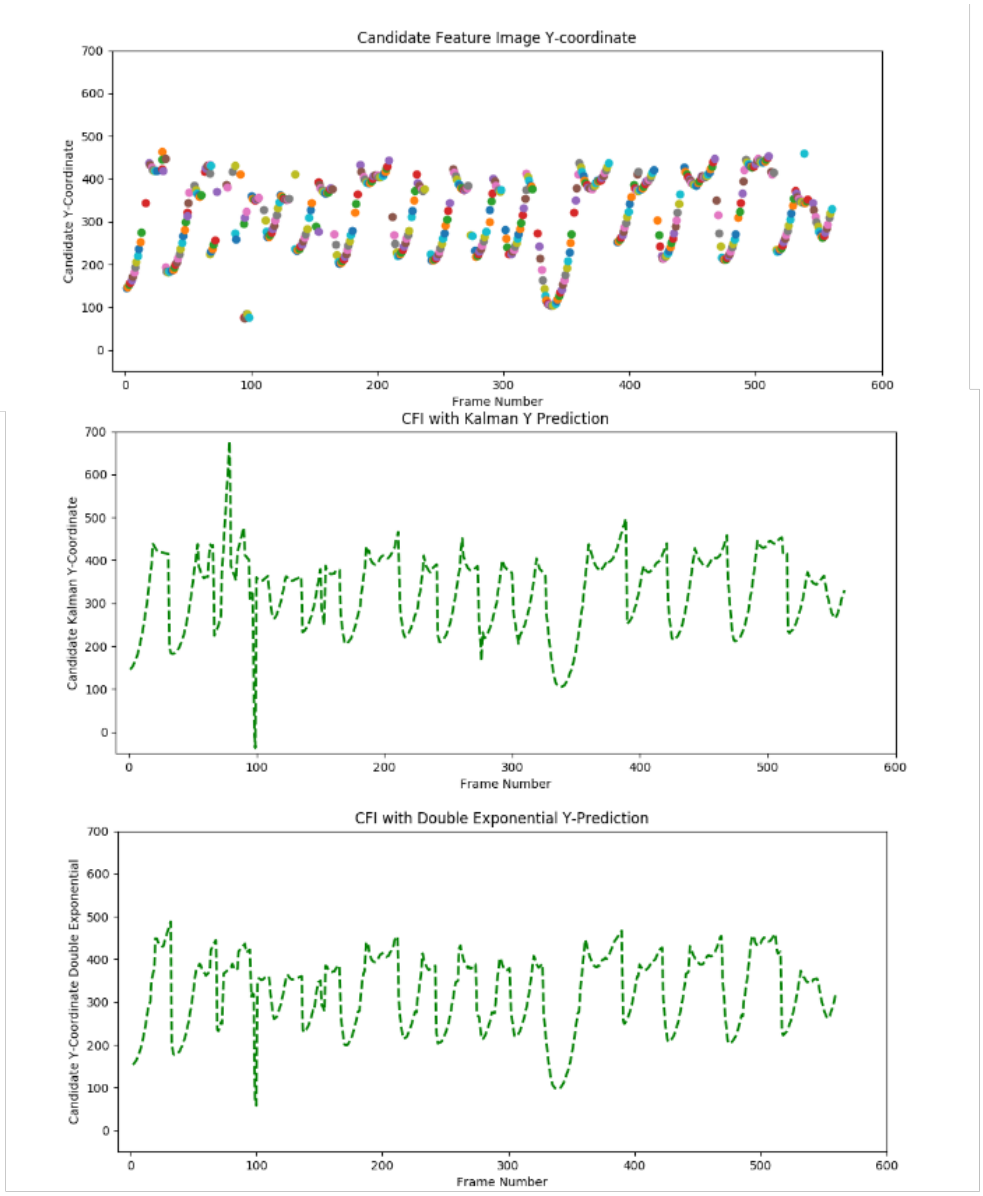


Figure 5.3: The Y-coordinate of the detected ball candidates plotted per frame (above) vs the trajectory formed using the Kalman Filter (middle) vs the trajectory formed using Double-Exponential Smoothing (below) for Dataset 1

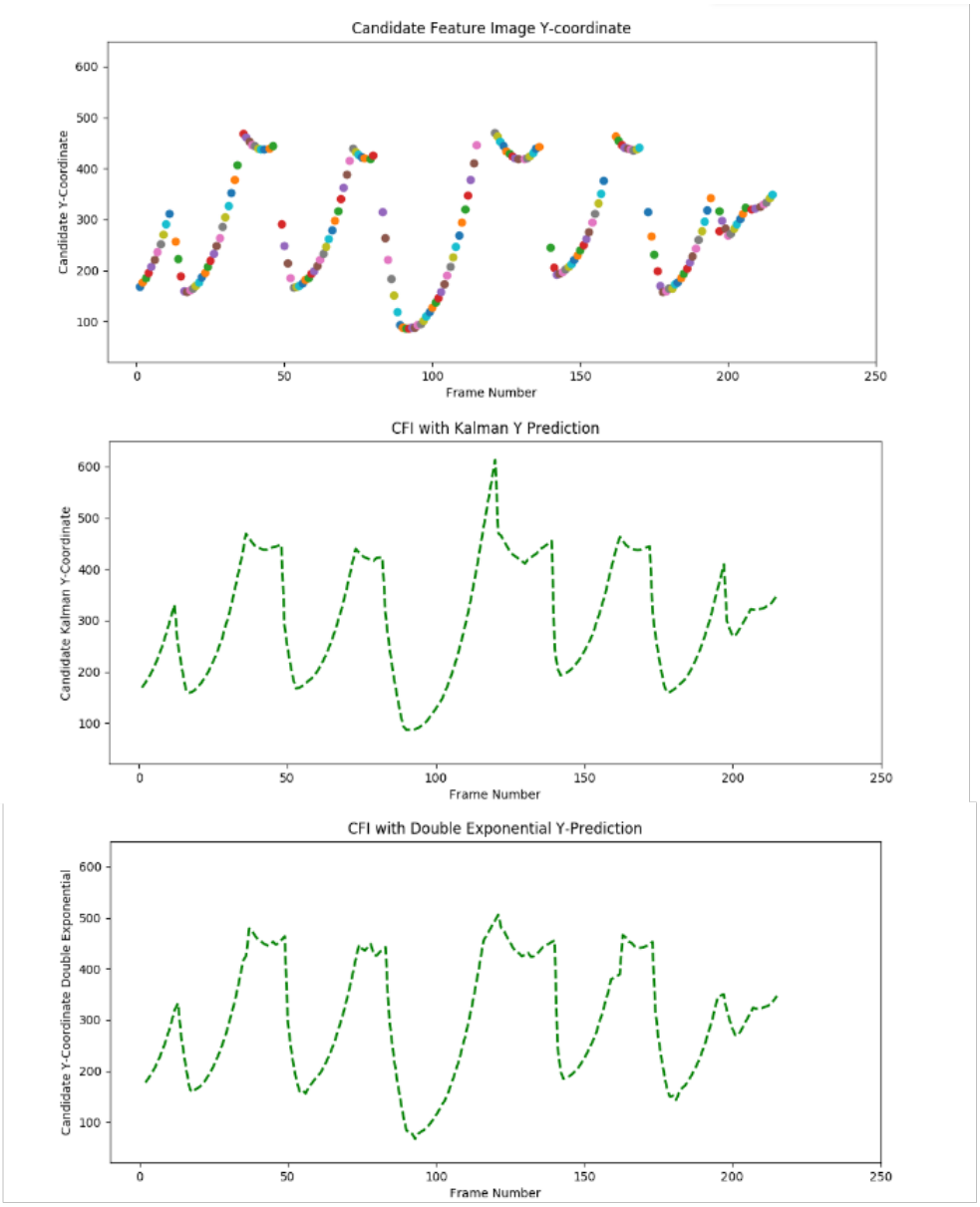


Figure 5.4: The Y-coordinate of the detected ball candidates plotted per frame (above) vs the trajectory formed using the Kalman Filter (middle) vs the trajectory formed using Double-Exponential Smoothing (below) for Dataset 2

good detection percentage (like Dataset 2) tend to have better final trajectories compared to the datasets which have low detection percentage (for instance Dataset 5).

5.4 CONCLUSION

The ball-tracking processes utilized in this thesis were discussed in this section. First, the popular method in the literature, the Kalman Filter, was utilized to perform tracking. Then, a method for forecasting in a time-series which is primarily used in financial applications, Holt's double-exponential smoothing, was utilized for tracking. Although, a quantitative score cannot be calculated to compare the exact accuracy of these methods due to the lack of ground-truth, a qualitative assessment shows that the exponential smoothing method results in more smoother and accurate 2D trajectories as compared to the Kalman Filter.

This approach indicates that using a straightforward recursive linear method can deliver better results compared to the more mathematical and complex method involving matrix calculations. This is a unique contribution to the ball-tracking problem, as this method hasn't previously been used to solve the problem in the literature (Chapter 2). As the exponential smoothing method has relatively lesser calculations compared to the Kalman method, it is also slightly faster in execution, as will be observed with timing analysis in Chapter 6.

6 OPTIMIZATIONS ON THE PROCESSOR

The ball-tracking procedures discussed in the previous chapters have been carried out on a Raspberry-Pi (RPI) model 3B+, which serves as a means to a low-cost approach for tracking. The software has been developed using Python3 and the open-source computer vision library (Bradski, 2000).

The RPI consists of a 1.4 GHz Arm Cortex-A53 processor and has 1 GB of RAM. The Cortex-A53 is a 64-bit quad-core processor with support for both 32-bit and 64-bit operations. It is built using the Arm v8 instruction set and carries over a lot of features from the Arm v7 architecture as can be seen in Figure 6.1. It contains support for floating point precision in the form of the VFPv3 (Vector Floating Point) instructions and for NEON advanced SIMD (Single Instruction Multiple Data) instruction set.

The dataset videos used in this project are of a 720p resolution with a frame-rate of 25 frames-per-second. This implies, for processing the videos under real-time constraints, each frame must be processed under 40 milliseconds. From initial timing analysis (before any optimization) of the execution times of the different processes in Table 6.1, it can be observed that the pre-processing stage is the bottleneck for the complete process as it takes the maximum amount of execution time. This is predictable as the operations in this stage takes place over three 720p input image frames which are represented by large arrays of pixel values. Another observation which can be made from Table 6.1 is about the execution times of the two methods used in the ball-tracking stage. It can be realized that the execution time of the Double-Exponential smoothing method is almost half of that of the Kalman filter method. But, since both of these execution times are lesser and incomparable to the execution-time of the pre-processing stage, the main focus is to optimize the pre-processing stage so that the complete process can function under real-time constraints.

There are two methods to optimize the process, 1) by utilizing the hardware capabilities and features of the processor, or 2) by using software techniques for faster processing. The hardware-based techniques include utilizing the features of the

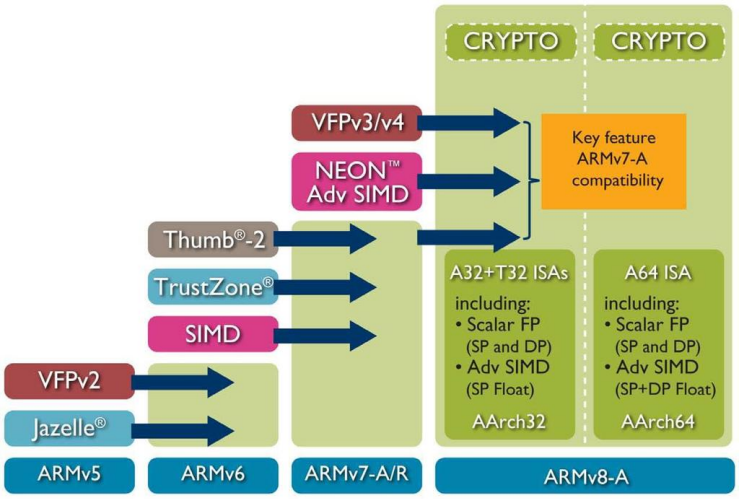


Figure 6.1: The evolution of Arm Architectures

Table 6.1: Timing results for processes before optimization – in milliseconds

Dataset	Pre-Processing	Detection	Tracking (Kalman Filter)	Total	Tracking (Double-Exponential)	Total
Dataset 1	662.31	22.94	3.43	724.23	1.58	722.38
Dataset 2	711.18	26.26	2.58	779.72	1.29	778.43
Dataset 3	640.52	25.27	2.94	705.08	1.49	703.63
Dataset 4	605.6	22.5	2.43	663.63	1.01	662.21
Dataset 5	621.38	23.54	2.61	681.18	1.01	679.58
Dataset 6	618.74	21.08	2.81	676.82	1.53	675.54
Dataset 7	619.01	21.84	2.84	676.97	1.62	675.75
Dataset 8	617.05	21.76	2.32	674.89	1.19	673.76
Dataset 9	601.5	21.31	2.71	659	1.52	657.81
Dataset 10	619.55	22.93	2.31	679.25	1.14	678.08

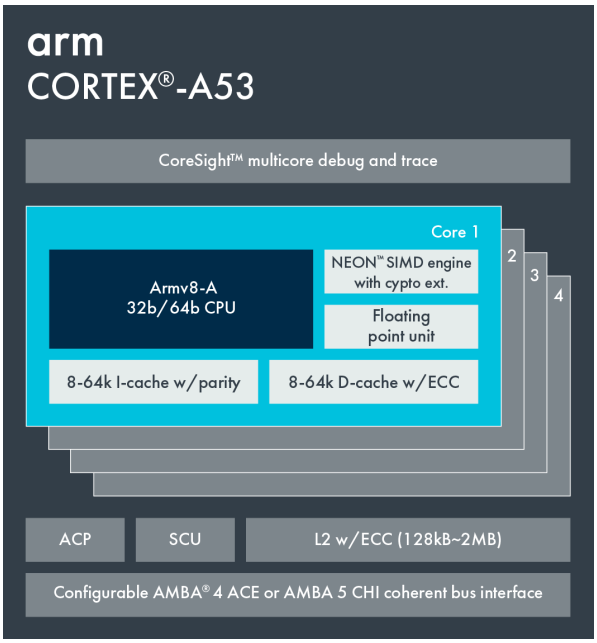


Figure 6.2: Arm Cortex A-53 Arm (2016)

Cortex processor such as its multiple cores, the advanced SIMD NEON instruction set and the VFPv3 floating-point support. Whereas, the software methods include adjusting the resolution or the frame-rate of the video to process it faster and using computer-vision functions that are fast and memory-efficient. Both of these approaches have been used in combination for optimization as discussed in the following sections.

6.1 ARM NEON AND VFPV3

One of the key features that the A53 Cortex processor possesses is the support for advanced SIMD instructions and floating-point optimization through the NEON architecture extension and the VFPv3 support. The NEON architecture extension by Arm (2017) is specifically developed to increase signal processing for higher-end applications such as those of Machine Learning and Computer Vision. It is a single instruction multiple data instruction-set which increases the processing for mathematical operations. Since, an image is just a large matrix of pixel values, the NEON extension optimizes operations on images for faster processing.

Table 6.2: Optimization results for the complete process using the different tracking methods – Total Execution Time in milliseconds

Dataset	Without any optimization		With NEON & VFPv3		With NEON, VFPv3 & 480p resolution	
	Kalman	Double Exponential	Kalman	Double Exponential	Kalman	Double Exponential
Dataset 1	724.23	722.38	366.49	364.83	202.63	201.1
Dataset 2	779.72	778.43	394.02	393.1	208.63	206.75
Dataset 3	705.08	703.63	362.52	360.55	189.85	187.75
Dataset 4	663.63	662.21	333.38	332.07	165.11	163.85
Dataset 5	681.18	679.58	328.21	326.75	176.29	175
Dataset 6	676.82	675.54	327.75	326.34	177.66	176.4
Dataset 7	676.97	675.75	330.85	329.4	174.03	172.84
Dataset 8	674.89	673.76	334.77	333.5	179.18	178.06
Dataset 9	659	657.81	331.22	329.8	196.25	194.68
Dataset 10	679.25	678.08	335.62	334.35	183.69	182.47
Average	692.077	690.717	344.483	343.069	185.332	183.89

Vector floating point (VFP) provides floating-point support in Arm processors. Since, the Arm v8 has support for both 64-bit and 32-bit operations, the VFPv3 provides support for half, single and double precision floating-point values. Bulk of the image-processing operations such as the Gaussian filter (Section 3.2.2) utilize floating-point calculation and VFPv3 helps in optimizing such operations.

The open-source computer vision library which has been predominantly used for implementation of computer vision methods in this thesis, contains support for the NEON and VFPv3 instruction set architectures (OpenCV, 2017). This means that when compiling the opencv library on an Arm v8 system, the NEON and VFPv3 hardware optimizations can be enabled. The opencv functions can then utilize the optimizations for faster processing.

The results of these optimizations can be observed in Table 6.2. The values in the table are the average execution times per frame (in milliseconds). It can be observed that by utilizing the NEON and VFPv3 optimizations, there is a 49% reduction in execution times of the complete process.

6.2 VIDEO RESOLUTION

The resolution of the video processed for detection can be modified for faster processing. This is due to the fact that images are mathematically represented as matrices, with the number of rows and columns defined by the resolution of the image. The dataset utilized in this thesis consists of 720p resolution videos (Section 3.1) which means the image matrix has dimensions of 720x1280 pixels.

To optimize the process, the images are resized to a lower 480p resolution during processing which reduces the image matrix dimensions to 480x854 pixels. The thresholds calculated for detection process are revised according to the dimensions, which results in no effect on the accuracy of the system. This conversion to a lower-resolution helps especially in processing the images faster during the pre-processing stage. The results of this operation can be observed in Table 6.2. The values in the table are the average execution times per frame (in milliseconds). It can be observed that by processing a lower resolution video on the NEON and VFPv3 architecture extensions, the performance further increases by 45%.

Table 6.3: Timing results for processes after optimization – in milliseconds

Dataset	Pre-Processing	Detection	Tracking (Kalman Filter)	Total	Tracking (Double-Exponential)	Total
Dataset 1	117.92	12.45	3.33	202.63	1.8	201.1
Dataset 2	119.71	14.19	3.58	208.63	1.7	206.75
Dataset 3	108.95	12.32	3.74	189.85	1.64	187.75
Dataset 4	95.16	9.91	2.68	165.11	1.42	163.85
Dataset 5	101.96	11.73	2.09	176.29	0.8	175
Dataset 6	103	12.44	2.68	177.66	1.42	176.4
Dataset 7	100.22	10.66	2.66	174.03	1.47	172.84
Dataset 8	106.5	10.2	2.68	179.18	1.56	178.06
Dataset 9	115.77	11.37	2.96	196.25	1.39	194.68
Dataset 10	106.53	12.22	2.71	183.69	1.49	182.47

6.3 MULTIPROCESSING

The Cortex A-53 is a quad-core processor which means its four cores can be utilized to perform parallel tasks. In the pre-processing stage, as can be observed in Figure 3.1, three consecutive images are processed for forming a frame-differential image. A way to optimize this process could be to perform these operations in parallel and process each individual image using a separate core of the processor using the multiprocessing module in Python3 (Python3 Docs, 2018b).

Unfortunately, the raspberry-pi suffers from memory limitation during implementation. The 1GB RAM of the hardware isn't able to process three images parallelly and throws an "out-of-memory" error during execution. The attempt to parallelize the process using multiple threads dedicated to each separate image for processing also doesn't work due to python's interpreter limitation. Since, python's interpreter is not thread-safe, python's Global Interpreter Lock (GIL) grants access only to a single thread at a time to access python objects (Python3 Docs, 2018a), which inhibits parallel operations.

6.4 FRAME-RATE

Another video-parameter which can be modified for increasing the performance of the system is the frame-rate at which the video is processed. The videos in the dataset are recorded at 25 frames per seconds. The aforementioned optimization steps reduces the execution times close to 200 milliseconds. But, for real-time processing the execution time should be under 40 milliseconds. This means, the input video needs to be process at $1/5^{th}$ of the original frame-rate, i.e. at a rate of 5 FPS, for processing each frame under real-time constraints.

The performance of the ball tracking-stage at a lower FPS can be visualized in Figure 6.3 and Figure 6.4. It can be observed that when the video is processed at a lower-frame rate, it suffers from a lack of ball-detection information. Since, squash is such a high-paced sport, by capturing information in every 5^{th} frame, plenty of movement and activity goes unnoticed in the tracking-stage. This performance is not suitable for a tracking system, therefore, frame-rate of the video cannot be modified for making the system real-time.

6.5 CONCLUSION

The optimizations techniques that are feasible to reduce the execution time of the complete process are discussed in this chapter. Using NEON & VFPv3 instruction

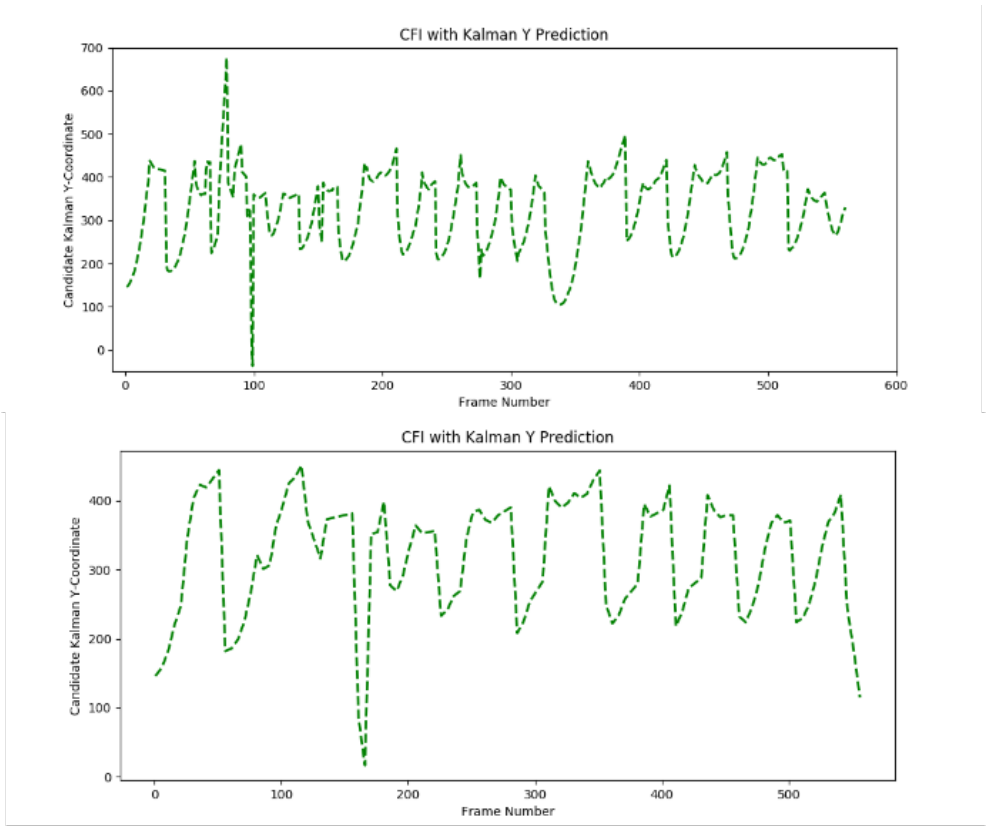


Figure 6.3: The Y-coordinates of the ball candidates plotted per frame at a frame-rate of 25 FPS (above) and 5 FPS (below) for Dataset 1

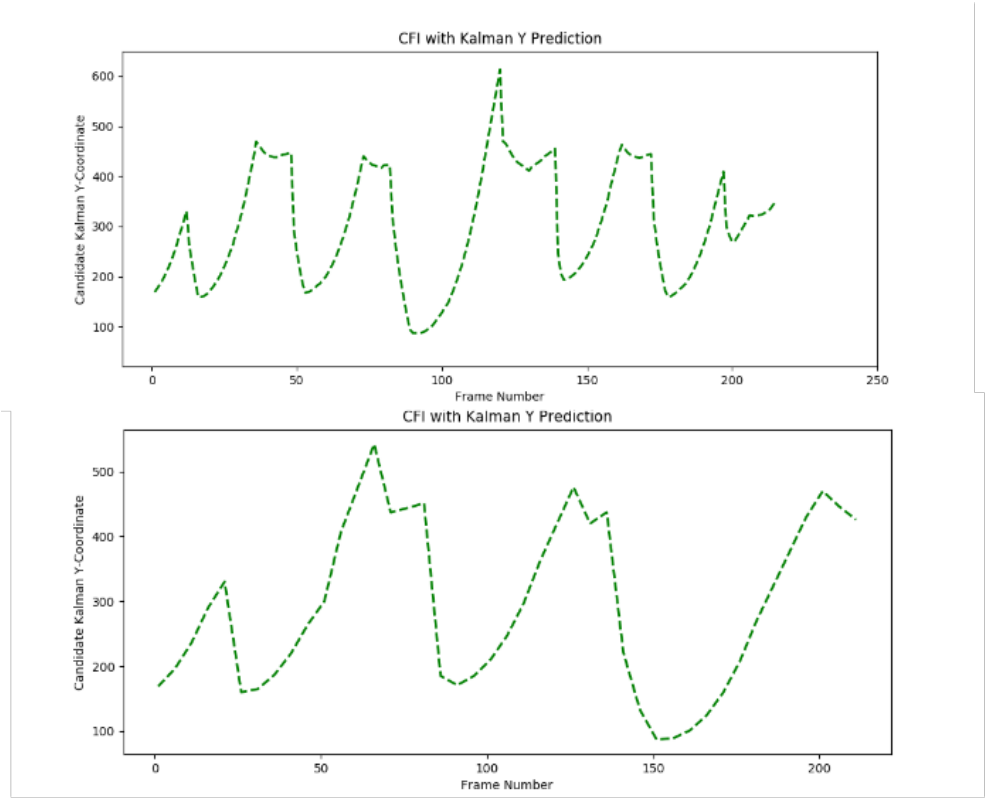


Figure 6.4: The Y-coordinates of the ball candidates plotted per frame at a frame-rate of 25 FPS (above) and 5 FPS (below) for Dataset 2

sets and processing the video at a 480p resolution, the execution times are reduced by almost 70%. But, other methods such as parallel processing and reducing the frame-rate are not found to be viable.

The aim was to optimize the procedure for real-time processing but as was hypothesized at the end of the literature study in Section 2.5, processing in real-time under such hardware-constraints is not feasible. From the timing analysis of the different processes after the optimization steps in Table 6.3, it can be observed that the pre-processing stage still takes the maximum amount of time to execute but this time has been drastically reduced from Table 6.1. This means, applications such as real-time line-calling during the game will not be feasible on a system built on a low-cost processor. But, since the system runs smoothly and outputs the trajectories without a huge delay, the low-cost system can be utilized adequately for game-analysis and shot-analysis after completion of the shot rally.

The limitations of the implementation on the processor are largely associated with the low-cost aspects of the hardware. A better hardware system with more memory and a better processor can rectify these limitations. A higher amount of RAM can result in parallel processing of the images in the pre-processing stage which can significantly reduce the execution-times. In terms of software, the limitation of multithreading in Python can be handled by using C++ instead of Python. As discussed in Section 6.3, Python only allows a single thread to run at an instance but C++ lets multiple threads to run at same time which can help run tasks concurrently.

7 CONCLUSION, DISCUSSION AND FUTURE WORK

7.1 SUMMARY

This research work is focused on creating a low-cost solution for ball-tracking in squash. Since, tennis as a sport bears a lot of similarity to squash, the tracking-systems built for tennis are first studied and evaluated. The detection methodologies and means of tracking the ball in tennis are assessed and the most suitable pre-processing, detection and tracking methods which could be effective for squash are hypothesized.

For the first stage of the process, a set of sequential operations are applied to the images in the processing pipeline. These operations result in a binary-image which is further used for detecting the ball.

The detection stage follows the pre-processing stage. A collection of detection methods inspired by the cues in the game are applied to detect the squash ball optimally. A varied set of squash matches are used as datasets to evaluate the accuracy of the detection methods. The detection stage results in an average accuracy of around 85%.

The detection stage is followed by the tracking stage, where the detections per frame in a dataset are combined to form a final ball-trajectory. A Kalman Filter and the Double-Exponential Smoothing methods are used as the two different methods for prediction in case of none or multiple ball-detections. The output trajectories are compared and evaluated qualitatively and the exponential smoothing method is observed to output better final trajectories.

The whole system is implemented on a Raspberry-Pi which functions as a low-cost system having an Arm Cortex-A53 processor chip and a limited amount of memory. The optimization steps taken to reduce the execution times are implemented and evaluated. A 70% decrease in processing time of the datasets is determined after the optimizations.

7.2 MAIN CONTRIBUTIONS

Ball-tracking has predominantly been performed in sports such as tennis, cricket, football and basketball. Even though squash is a widely popular sport with around 20 million active players (US Squash, 2018), but since it lacks mainstream popularity, ball-tracking solutions in squash haven't been developed. The only other previous work that attempts to track a squash-ball is Rozumnyi et al. (2017) which reports a 0% detection accuracy for squash. The reason for this is that the authors aim to build a universal system for detection and tracking of the complete class of "fast-moving objects." This generalized detection method doesn't fit to the peculiarities of squash and a *more* focused system is needed to solve the specific challenges of squash. This thesis builds a dedicated detection and tracking process which focuses on the specific challenges of squash.

This thesis answers the problem-statement specified in Section 1.2 by solving the specified objectives in the following manner:

- The ball-detection methods that are commonly used for the popular sports aren't successful in squash. The popular methods of color-based segmentation

and shape-analysis of objects to detect the ball are not favorable in squash due to the high-speed movement of the squash ball. Therefore, detection methods such as size-based segmentation, region-based filtering and velocity-based constraints are used for detecting a squash ball.

- The detection system built using these methods is tested on a variety of datasets. The dataset contains matches of both male and female players, at different squash courts and using both white and black colored balls. The F1-score has been used as the parameter to quantify the accuracy of the system. The system results in an average accuracy of 85%. The system is found to perform well in squash matches with courts having a dark background and a white ball, but performs unsatisfactorily in courts with a white-background. This is due to the inability to detect the semi-transparent streak of the squash ball over a light-background.

- The tracking method initially identified to be the most suitable for this application is the Kalman Filter due to its accurate results with a light memory footprint. Its implementation results in good results for common occlusion scenarios but performs inadequate tracking when the ball changes its trajectory. This is due to the prevalent linear nature of the filter.

- The Double Exponential Smoothing which takes into account the trend of the data is used to rectify the above-mentioned problem of the Kalman filter. It is a novel method for ball-tracking as it hasn't been used in this domain. It's a fairly simple but efficient approach for prediction which uses linear and recursive techniques to output the estimated ball position. The results of the two methods are compared qualitatively using a Candidate-feature image which shows that the Double-exponential method results in better ball-trajectories than the Kalman filter.

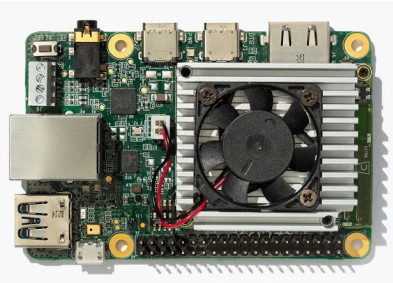
- A combination of software and hardware optimization steps are devised to enhance the performance of the system on a low-cost processor. Arm architecture extensions NEON and VFPv3 have been utilized along with the alteration of the resolution of the input videos to optimize the processing. A timing analysis of the complete process shows a reduction of almost 70% in the execution time of the complete process after using these optimization steps.

- In the end though, the optimization isn't enough for the system to process under real-time constraints but is fast and accurate enough to output smooth 2D ball-trajectories without a large delay. This means, application such as real-time line calling can not work with this system but other applications such as game-analysis and shot-analysis post the completion of the shot-rally can be utilized with this system.

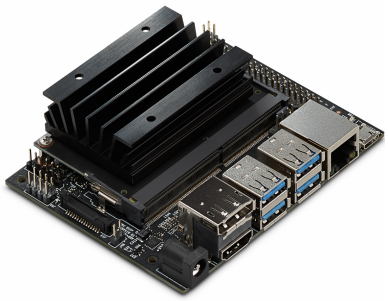
The objectives specified at the start of the research-work are all successfully met. This thesis work is a first of its kind to tackle the problems of ball-tracking in squash. The main contributions of this thesis are:

1. Since the only work before this thesis which aimed to detect a squash ball reports insignificant results, this thesis results in a primal and distinctive work for ball-tracking in squash. It aims to cater to the needs of the 20 million squash players that actively play this sport.

2. This work augments the use of detection methods which focus on removing noisy candidates by their region or their velocity. This is in contrast to the popular strategies which use color and shape for filtering out the false candidates. This work shows that using velocity and region based constraints can provide optimal results in detection accuracy.



(a) Google Coral Development Board Google (2019)



(b) NVIDIA Jetson Nano NVIDIA (2019)

Figure 7.1: Low-cost development boards with support for machine learning applications

3. This thesis applies a novel method to perform ball-tracking which produces better results as compared to the traditional method used for tracking. The Double-Exponential Smoothing technique is a simple yet powerful method which has been primarily used for prediction in time-series' for financial applications. Since, the squash ball follows an approximately linear trajectory, the coordinates of the the ball can be predicted optimally using the smoothing method. The double exponential method takes into account the trend of the data for its prediction, which means if there is a change in the direction of the ball, this method can predict the position of the ball much more efficiently compared to the Kalman Filter.

4. This thesis researches into the utilization of low-cost techniques such as a single camera view and a low-cost processor to perform the ball-tracking process. This is a novel approach to perform ball-tracking which is in contrast to the approach currently used for mainstream sports such as tennis where there are a number of cameras and high-computing resources to perform this task. The work in this thesis demonstrates that ball-tracking process can be performed in a low-cost manner given that certain optimization steps are performed to increase the processing. With the advent of high-processing power at lower costs, this thesis furthers the idea of utilization of such a low-cost approach over the current high-cost state-of-the-art methods to track the ball.

7.3 FUTURE WORK

The future scope of this project lies in utilizing the techniques of Deep-Learning in combination with the computer vision techniques to perform ball-detection with additional accuracy. The accuracy of the ball-detection process plays a major role in generating an accurate ball-trajectory. An image classifier built using convolutional neural networks as built for soccer in Kamble et al. (2019) can be utilized for detecting the ball-candidates.

To do so with a low-cost processor still remains a challenge. An approach could be to build custom hardware which supports the required processing as done in In/Out (2017). Another approach could be to utilize the NVIDIA Jetson nano (NVIDIA, 2019) or the Google Coral (Google, 2019) development boards, which were launched this year to build dedicated edge-computing applications involving artificial intelligence (Figure 7.1). At a price point of 100 - 150 USD, they can be aptly utilized for a low-cost solution.

A future extension of this project could be to build the complete process on a dedicated unit which houses the processor and the camera together. There are a few advantages for such a solution:

1. A calibration system can be built on such a solution, which calculates the boundary of the squash courts and uses the camera parameters to define the size based-thresholds used in the detection process. In this thesis, these values have been calculated using a statistical analysis but on a dedicated system these values can be derived automatically.

2. The portable nature of the solution means that it can be used on various squash courts for both amateur and professional squash matches. This will resolve one of the limitations of this thesis as the data used in this thesis has been obtained exclusively from professional squash matches since it is difficult to obtain data from amateur matches.

3. Such a dedicated system enables the use of an improved camera setup with higher frame-rates. As was observed before, a better frame-rate can aid in detecting the squash ball which helps in building improved final ball trajectories.

BIBLIOGRAPHY

Archana, M. and Geetha, M. K. (2015). Object detection and tracking based on trajectory in broadcast tennis video. *Procedia Computer Science*, 58:225–232. 6, 7, 9, 10

Arm (2016). Cortex-A53. https://developer.arm.com/ip-products/processors/cortex-a/cortex-a53. Accessed: 2019-06-12. xii, 56

Arm (2017). Neon ISA. https://developer.arm.com/architectures/instruction-sets/simd-isas/neon. Accessed: 2019-06-12. 56

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 3, 20, 21, 32, 55

Brown, R. G. (1957). Exponential smoothing for predicting demand. In *Operations Research*, volume 5, pages 145–145. INST OPERATIONS RESEARCH MANAGEMENT SCIENCES 901 ELKRIDGE LANDING RD, STE 49

Chakraborty, B. and Meher, S. (2012a). Real-time position estimation and tracking of a basketball. In *2012 IEEE International Conference on Signal Processing, Computing and Control*, pages 1–6. IEEE. 13

Chakraborty, B. and Meher, S. (2012b). A trajectory-based ball detection and tracking system with applications to shot-type identification in volleyball videos. In *2012 International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5. IEEE. 11, 13

Chen, B. and Wang, Z. (2007). A statistical method for analysis of technical data of a badminton match based on 2-d seriate images. *Tsinghua science and technology*, 12(5):594–601. 13

Conaire, C. Ó., Kelly, P., Connaghan, D., and O'Connor, N. E. (2009). Tennissense: A platform for extracting semantic information from multi-camera tennis data. In *Digital Signal Processing, 2009 16th International Conference on*, pages 1–6. IEEE. xi, 5, 6, 8, 10

Dilation (2019). Dilation and Erosion by Mathworks. https://nl.mathworks.com/help/images/morphological-dilation-and-erosion.html. Accessed: 2019-03-08. xi, 27

Dougherty, E. R. and Lotufo, R. A. (2003). *Hands-on morphological image processing*, volume 59. SPIE press. 27

Ekinci, B. D. and Gokmen, M. (2008). A ball tracking system for offline tennis videos. In *Proceedings of the 1st WSEAS international conference on Visualization, imaging and simulation*, pages 45–48. World Scientific and Engineering Academy and Society (WSEAS). 6, 7, 9, 10, 11

Erosion (2019). Morphological Erosion Visualization. www.cs.princeton.edu/~pshilane/class/mosaic/. Accessed: 2019-03-08. xi, 28

Esme, B. (2009). Kalman filter for dummies. *Bilgin's Blog, Mar*. xii, 48

Fazio, M., Fisher, K., and Fujinami, T. (2018). Tennis ball tracking: 3d trajectory estimation using smartphone videos. xi, 6, 7, 9, 10, 11

Forbes (2003). Squash: Number one Healthiest Sport to play. https://www.forbes.com/2003/09/30/cx_ns_1001featslide.html?thisSpeed=20000#2f0602e35ba6. Accessed: 2019-03-25. 1

Forsyth, D. A. and Ponce, J. (2003). A modern approach. *Computer vision: a modern approach*, 17:21–48. 32

Gabriel, P. F., Verly, J. G., Piater, J. H., and Genon, A. (2003). The state of the art in multiple object tracking under occlusion in video sequences. In *Advanced Concepts for Intelligent Vision Systems*, pages 166–173. xi, 14, 15

Google (2019). Coral. https://coral.withgoogle.com/. Accessed: 2019-06-12. 63

Harris, C. G., Stephens, M., et al. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer. 32

Hawk-Eye Innovations (2019). 3D Ball Tracking with Hawk-Eye Innovation. https://www.hawkeyeinnovations.com/. Accessed: 2019-05-27. xi, 1

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10. 45, 49

Hrabalík, A. (2017). Implementing and applying fast moving object detection on mobile devices. 7

Huang, M. Y.-k. and Huang, C.-p. (2017). An optimal image segmentation clustering algorithm for badminton sport moving tracking applications from video sequences. *IJCSIS*. xi, 6, 9

Huang, Y., Llach, J., and Zhang, C. (2008). A method of small object detection and tracking based on particle filters. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE. 12

Ian McKenzie (2017). Squash Balls. http://www.squashplayer.co.uk/squash_balls.htm. Accessed: 2019-04-11. 31

In/Out (2017). In/Out The Portable Ready-to-Use Line Call Device. https://inout.tennis/. Accessed: 2019-01-18. 1, 63

Kamble, P., Keskar, A., and Bhurchandi, K. (2019). A deep learning ball tracking system in soccer videos. *Opto-Electronics Review*, 27(1):58–69. 63

Kamble, P. R., Keskar, A. G., and Bhurchandi, K. M. (2017). Ball tracking in sports: a survey. *Artificial Intelligence Review*, pages 1–51. 2, 6, 9, 10, 11, 12, 13, 14

Kanan, C. and Cottrell, G. W. (2012). Color-to-grayscale: does the method matter in image recognition? *PloS one*, 7(1):e29740. 20

Kim, J.-Y. and Kim, T.-Y. (2009). Soccer ball tracking using dynamic kalman filter with velocity control. In *2009 Sixth International Conference on Computer Graphics, Imaging and Visualization*, pages 367–374. IEEE. xi, 11, 12

Mao, J., Mould, D., and Subramanian, S. (2007). Background subtraction for realtime tracking of a tennis ball. In *VISAPP (2)*, pages 427–434. Citeseer. 7, 9, 10

NVIDIA (2019). Jetson Nano. https://developer.nvidia.com/embedded/jetson-nano-developer-kit. Accessed: 2019-06-12. 63

OpenCV (2017). CPU optimizations build options. https://github.com/opencv/opencv/wiki/CPU-optimizations-build-options. Accessed: 2019-06-12. 57

Owens, N., Harris, C., and Stennett, C. (2003). Hawk-eye tennis system. In *Visual Information Engineering, 2003. VIE 2003. International Conference on*, pages 182–185. IET. 1, 6, 9, 12, 15

Pingali, G., Opalach, A., and Jean, Y. (2000). Ball tracking and virtual replays for innovative tennis broadcasts. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 152–156. IEEE. 6, 8, 9, 12

Pingali, G. S., Jean, Y., and Carlbom, I. (1998). Real time tracking for enhanced tennis broadcasts. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 260–265. IEEE. 6, 8, 9

Polceanu, M., Petac, A.-O., Lebsir, H. B., Fiter, B., and Buche, C. (2018). Real time tennis match tracking with low cost equipment. In *FLAIRS-31*, pages 197–200. 5, 6, 13

Python3 Docs (2018a). Global Interpreter Lock. https://docs.python.org/3/glossary.html#term-global-interpreter-lock. Accessed: 2019-06-12. 58

Python3 Docs (2018b). Process-based parallelism. https://docs.python.org/3/library/multiprocessing.html. Accessed: 2019-06-12. 58

Qazi, T., Mukherjee, P., Srivastava, S., Lall, B., and Chauhan, N. R. (2015). Automated ball tracking in tennis videos. In *Image Information Processing (ICIIP), 2015 Third International Conference on*, pages 236–240. IEEE. 5, 9, 10, 13

Rozumnyi, D., Kotera, J., Sroubek, F., Novotnỳ, L., and Matas, J. (2017). The world of fast moving objects. In *CVPR*, pages 4838–4846. 2, 8, 9, 10, 18, 61

Shruti Saxena (2018). Precision vs Recall. https://towardsdatascience.com/precision-vs-recall-386cf9f89488. Accessed: 2019-04-29. 40

Sonka, M., Hlavac, V., and Boyle, R. (1993). *Image pre-processing*, pages 56–111. Springer US, Boston, MA. 17

Squash TV (2019). PSA Squash TV. https://psaworldtour.com/tv. Accessed: 2019-03-06. 18

Suzuki, S. et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46. xi, 32, 33

Teachabarikiti, K., Chalidabhongse, T. H., and Thammano, A. (2010). Players tracking and ball detection for an automatic tennis video annotation. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 2461–2494. IEEE. xi, 6, 8, 9, 10

US Squash (2018). Squash Facts. https://www.ussquash.com/squash-facts/. Accessed: 2019-06-12. 61

Welch, G., Bishop, G., et al. (1995). An introduction to the kalman filter. 47

Yan, F., Christmas, W., and Kittler, J. (2005). A tennis ball tracking algorithm for automatic annotation of tennis match. In *British machine vision conference*, volume 2, pages 619–628. 5, 11, 12, 13

Yu, X., Sim, C.-H., Wang, J. R., and Cheong, L. F. (2004). A trajectory-based ball detection and tracking algorithm in broadcast tennis video. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 2, pages 1049–1052. IEEE. xi, 4, 6, 9, 10, 11, 12, 13, 16, 50

Zhou, X., Xie, L., Huang, Q., Cox, S. J., and Zhang, Y. (2015). Tennis ball tracking using a two-layered data association approach. *IEEE Transactions on Multimedia*, 17(2):145–156. xi, 6, 10, 11, 13, 14, 37

A DETECTION ALGORITHMS

Algorithm A.1: Detecting Candidates using Size

Output : *BallCandidates*, *PlayerCandidates*, *IncompletePlayerCandidates*
Input : *ListOfContours*, *CurrentFrame*
Initialize : *minBallArea*, *maxBallArea*, *minPlayerArea*

```
for contour in ListOfContours do
    Compute the area of the contour;
    Compute the X,Y coordinates of the centre of the contour as cX, cY;
    if area > minPlayerArea then
        Update PlayerCandidates with the area, cX, cY, contour properties of
         the player candidate
    else if area < minPlayerArea and area > maxBallArea then
        Update IncompletePlayerCandidates with the area, cX, cY, contour
         properties of the incomplete player candidate
    else if area < maxBallArea and area > minBallArea then
        Update BallCandidates with the area, cX, cY, contour properties of
         the ball candidate
    else if area < minBallArea then
        continue
end
```

Algorithm A.2: Detecting Candidates Using Court Boundaries

Output : *ballCandidatesFiltered*, *playerCadidatesFiltered*, *incompletePlayerCandidatesFiltered*
Input : *dataset*, *ballCandidates*, *playerCadidates*, *incompletePlayerCandidates*, *currentFrame*
Initialize : *courtMinXCoordinate*, *courtMaxXCoordinate* (based on the *dataset*);
ballCandidatesFiltered as an empty List;
playerCadidatesFiltered as an empty List;
incompletePlayerCandidatesFiltered as an empty List;

```
for candidate in ballCandidates do
    if cX < courtMinXCoordinate or cX > courtMaxXCoordinate then
        continue
    else
        Update ballCandidatesFiltered with the candidate
    end
end
for candidate in playerCandidates do
    if cX < courtMinXCoordinate or cX > courtMaxXCoordinate then
        continue
    else
        Update playerCandidatesFiltered with the candidate
    end
end
for candidate in incompletePlayerCandidates do
    if cX < courtMinXCoordinate or cX > courtMaxXCoordinate then
        continue
    else
        Update incompletePlayerCandidatesFiltered with the candidate
    end
end
```

Algorithm A.3: Detecting Candidates using Player Proximity

```
Output    : ballCandidatesFiltered
Input     : ballCandidates, playerCadidates, incompletePlayerCandidates,
            currentFrame
Initialize: minBallDistance, minDistance;
ballCandidatesFiltered as an empty List
if ballCandidates is empty then
    print "No ball Candidates in the frame"
else
    Initialize minDist to a MAX value;
    for candidate in ballCandidates do
        if playerCandidates > 1 then
            for player in playeCandidates do
                Calculate Eucledian Distance dist between player and
                 candidate;
                if dist < minDist then
                    Update minDist with dist
            end
        else if playerCandidates == 1 then
            Set player as the only element in playerCandidates;
            Calculate Eucledian Distance dist between player and candidate;
            if dist < minDist then
                Update minDist with dist
            for playerPart in incompletePlayerCandidates do
                Calculate Eucledian Distance dist between playerPart and
                 candidate;
                if dist < minDist then
                    Update minDist with dist
            end
        else if incompletePlayerCandidates > 1 then
            for playerPart in incompletePlayerCandidates do
                Calculate Eucledian Distance dist between playerPart and
                 candidate;
                if dist < minDist then
                    Update minDist with dist
            end
        else
            continue
        if minDist ≥ minBallDistance then
            Update ballCandidatesFiltered with the candidate
    end
```

Algorithm A.4: Detecting Candidates using their Motion

```
   Output    : ballCandidatesFiltered, ballCandidatesPeviousFrame
   Input     : ballCandidates, ballCandidatesPreviousFrame, CurrentFrame
   Initialize: minMotionDistance, maxMotionDistance;
   ballCandidatesFiltered as an empty List
 1 if ballCandidatesPreviousFrame > o then
 2 |   for candidate in ballCandidates do
 3 |   |   Set ballCandidateFlag = False;
 4 |   |   for previousCandidate in ballCandidatesPreviousFrame do
 5 |   |   |   Calculate Eucledian Distance dist between candidate and
   |   |   |    previousCandidate;
 6 |   |   |   if dist > minMotionDistance and dist < maxMotionDistance then
 7 |   |   |   |   set ballCandidateFlag as True
 8 |   |   |   else
 9 |   |   |   |   continue
10 |   |   end
11 |   |   if ballCandidateFlag is True then
12 |   |   |   Update ballCandidatesFiltered with the candidate
13 |   |   else
14 |   |   |   continue
15 |   end
16 else
17 |   Copy candidates from ballCandidates to ballCandidatesFiltered
18 Copy candidates from ballCandidates in ballCandidatePreviousFrame
```

B BALL TRACKING RESULTS

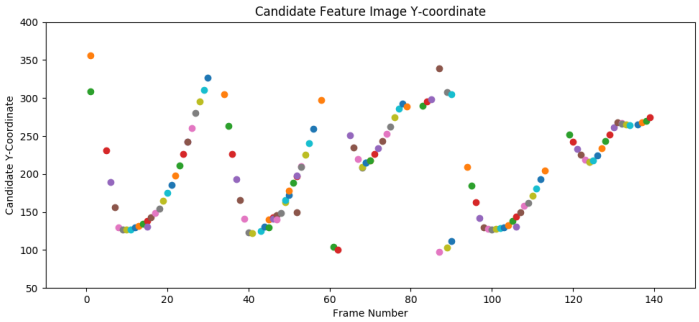


Figure B.1: The Y-coordinate of the detected ball candidates plotted per frame for Dataset3

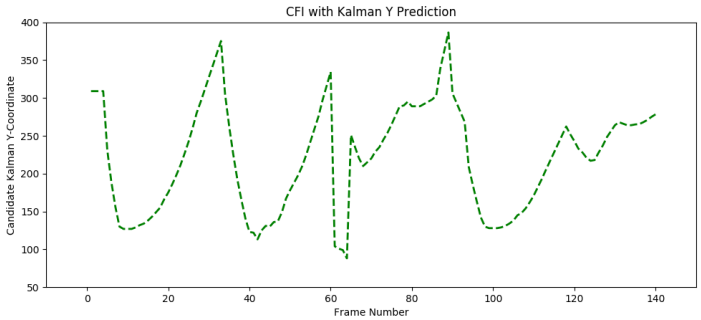


Figure B.2: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset3

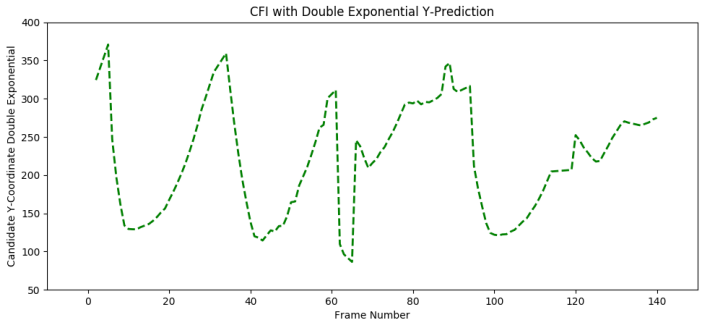


Figure B.3: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset3

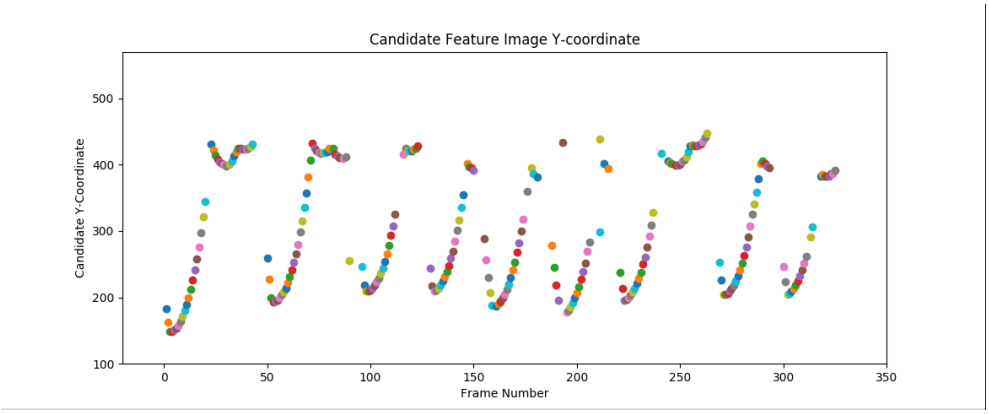


Figure B.4: The Y-coordinate of the detected ball candidates plotted per frame for Dataset4

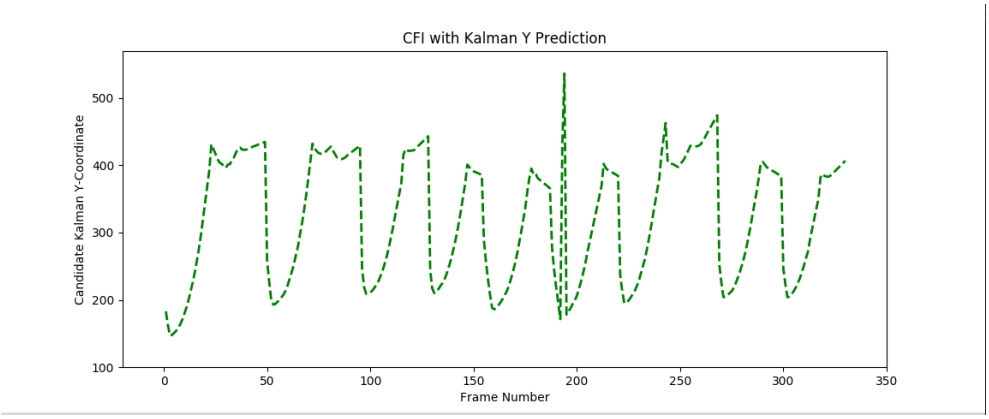


Figure B.5: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset4

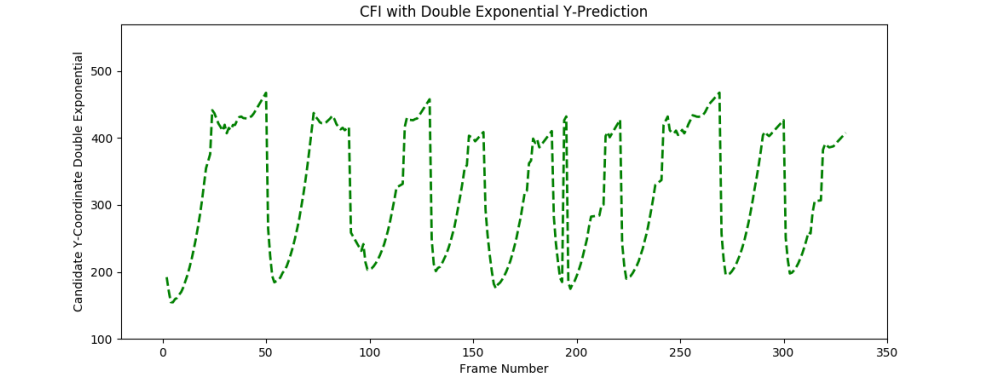


Figure B.6: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset4

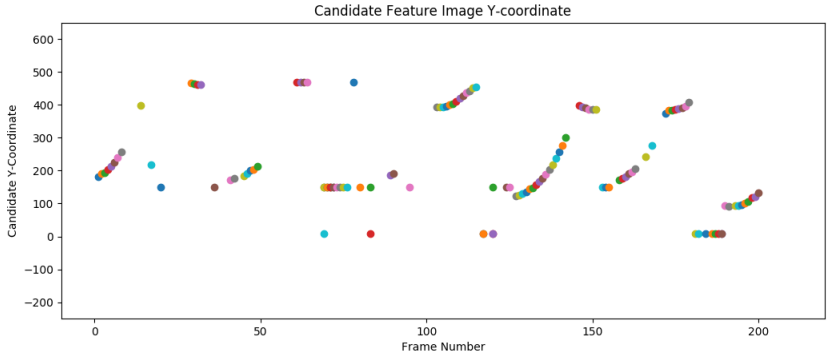


Figure B.7: The Y-coordinate of the detected ball candidates plotted per frame for Dataset5

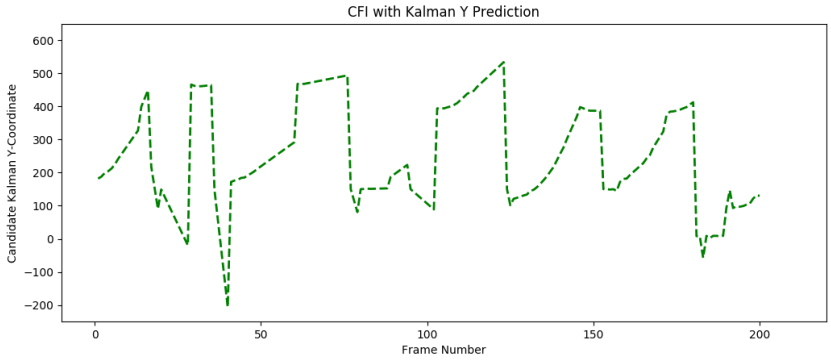


Figure B.8: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset5

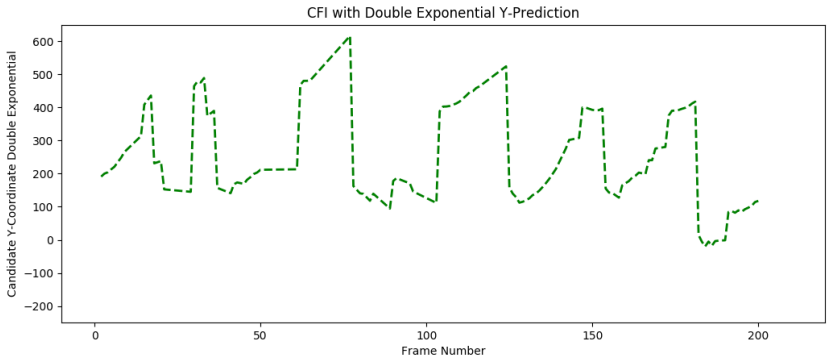


Figure B.9: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset5

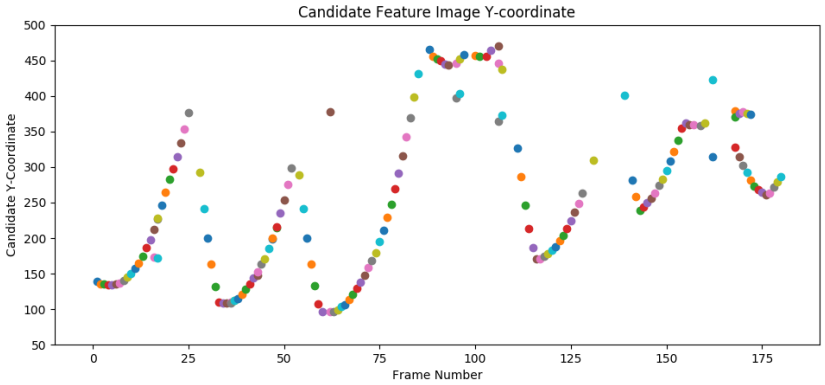


Figure B.10: The Y-coordinate of the detected ball candidates plotted per frame for Dataset6

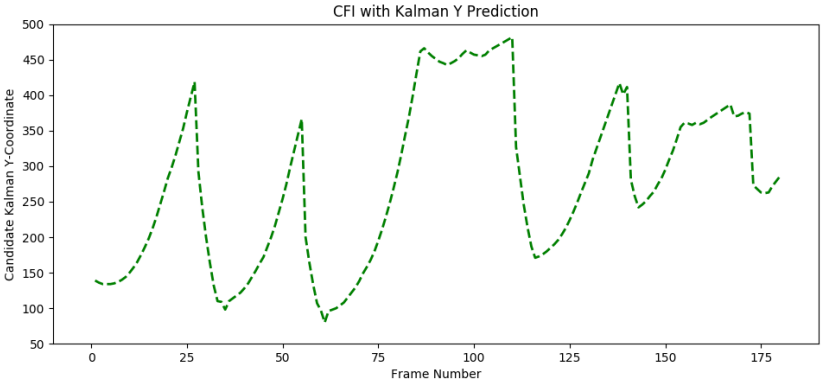


Figure B.11: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset6

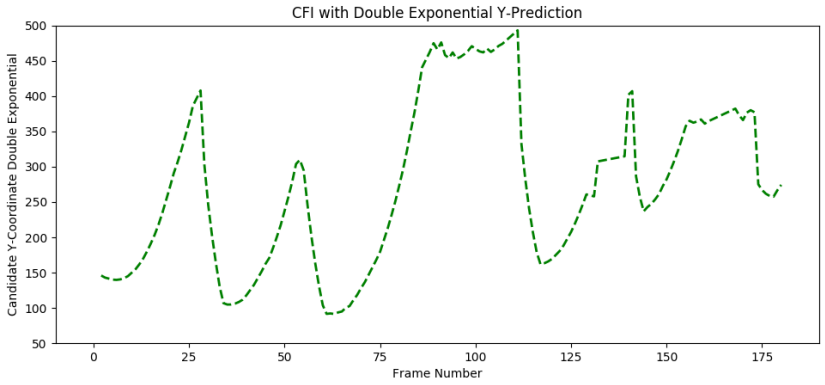


Figure B.12: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset6

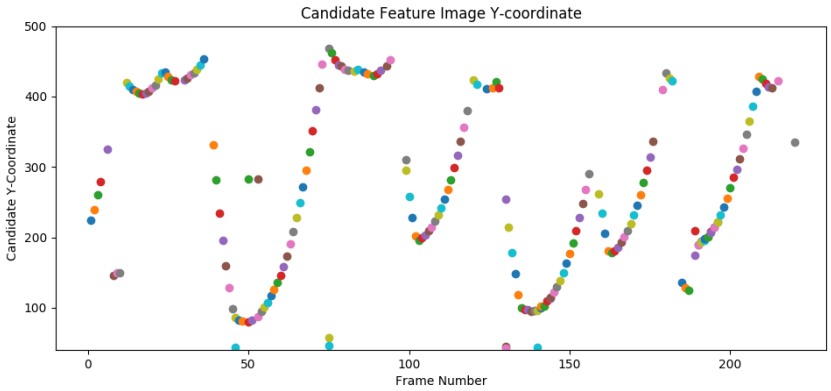


Figure B.13: The Y-coordinate of the detected ball candidates plotted per frame for Dataset7

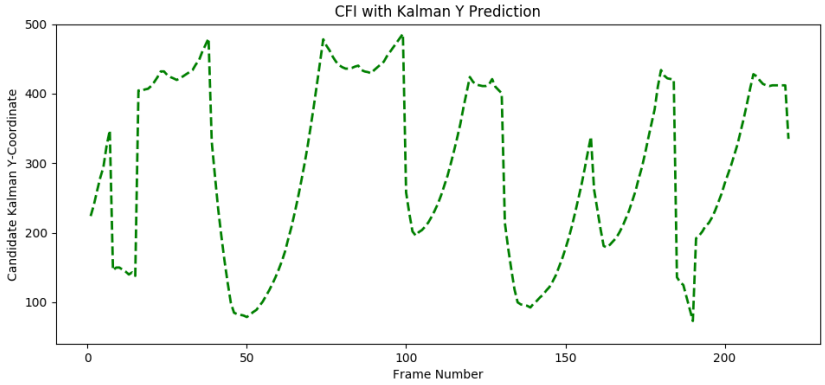


Figure B.14: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset7

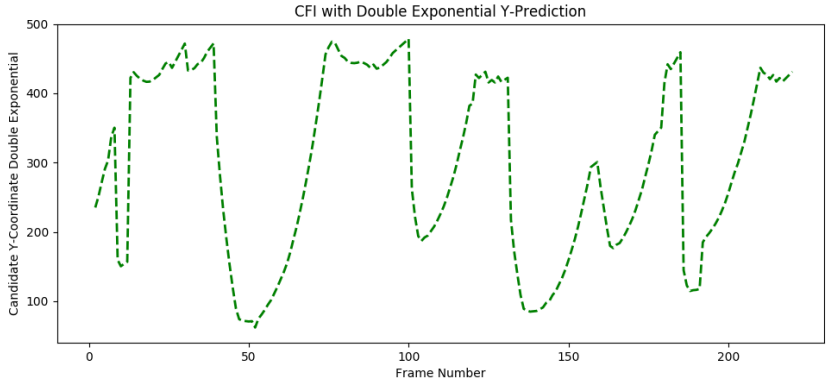


Figure B.15: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset7

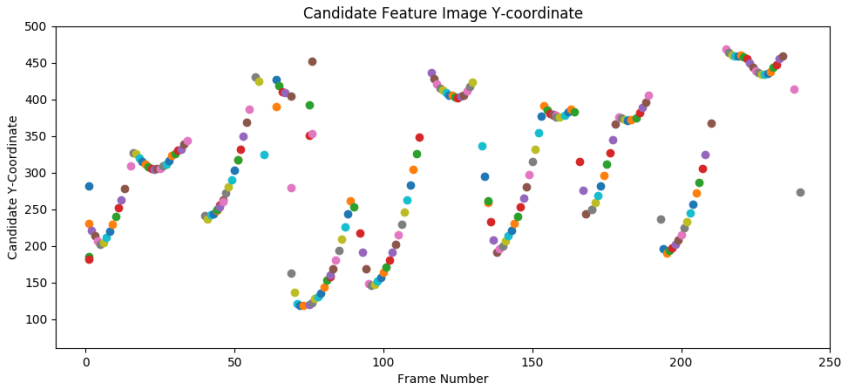


Figure B.16: The Y-coordinate of the detected ball candidates plotted per frame for Dataset8

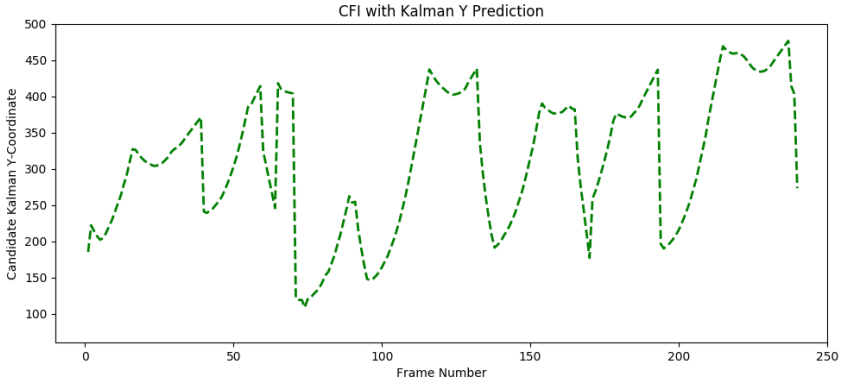


Figure B.17: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset8

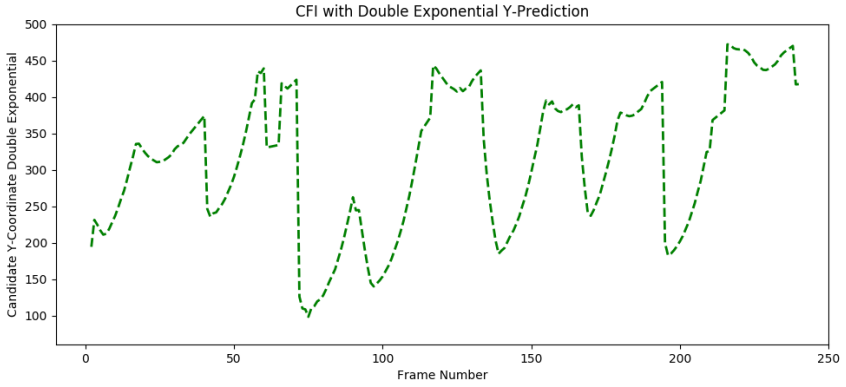


Figure B.18: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset8

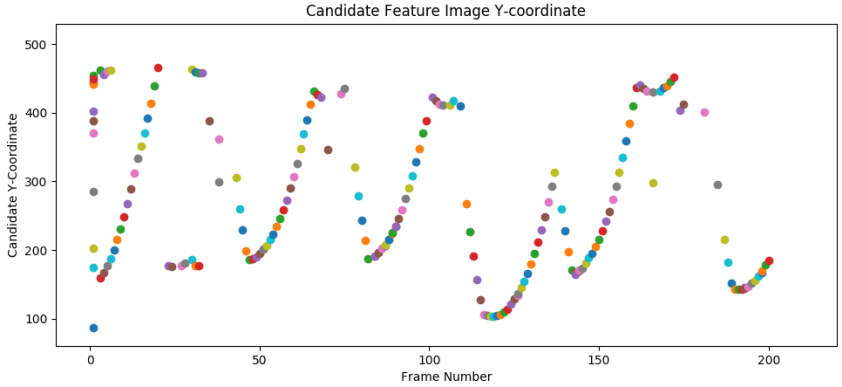


Figure B.19: The Y-coordinate of the detected ball candidates plotted per frame for Dataset9

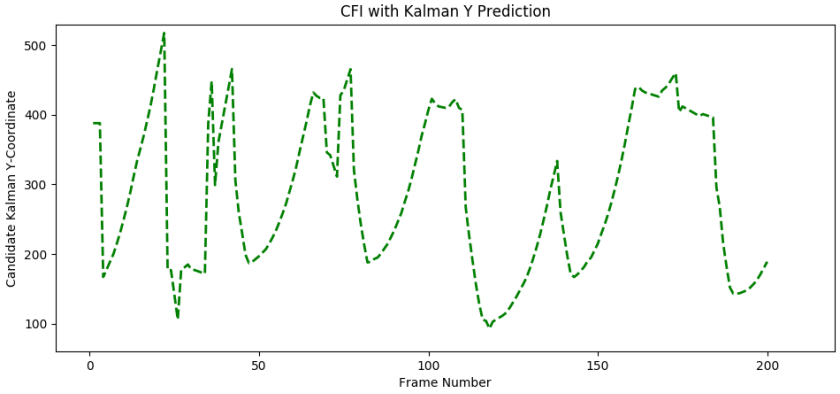


Figure B.20: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset9

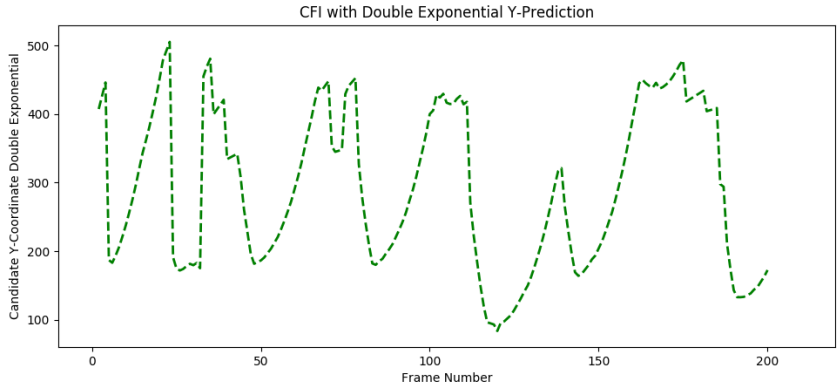


Figure B.21: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset9

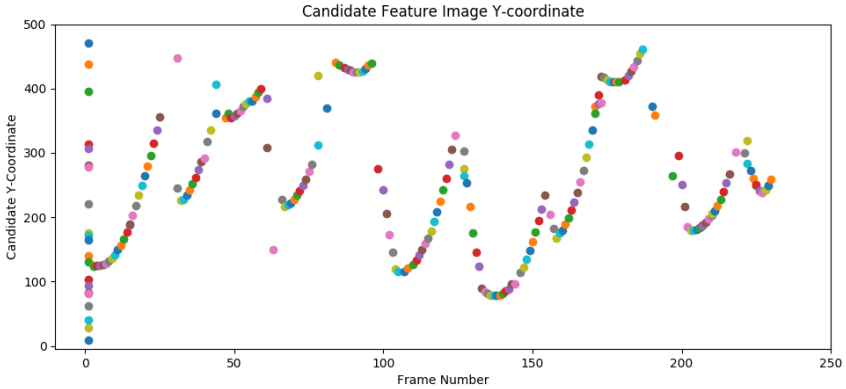


Figure B.22: The Y-coordinate of the detected ball candidates plotted per frame for Dataset10

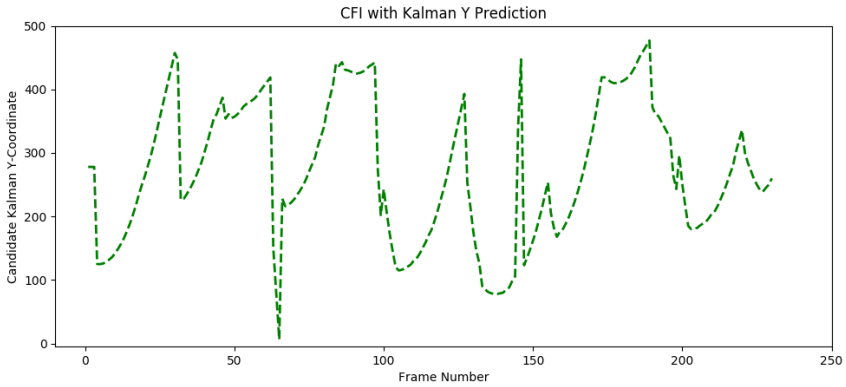


Figure B.23: The Y-coordinates of the ball candidates plotted per frame using the Kalman Filter approach for Dataset10

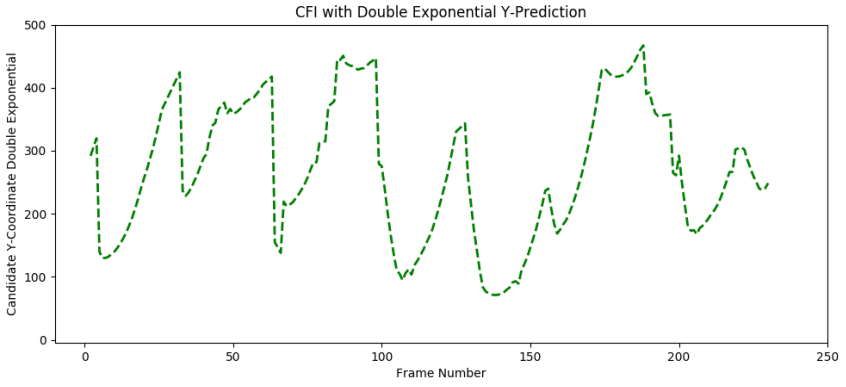


Figure B.24: The Y-coordinates of the ball candidates plotted per frame using the Double-exponential smoothing approach for Dataset10