Waterford Institute of Technology
School of Science

Basketball Shot Detection with Deep Learning

Waterford Institute *of* Technology

INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Emmanuel Sedicol
20072377

Supervisor: Pat McInerny

A report submitted in partial fulfilment of the requirements
for the degree of
Bachelor of Science (Honours) in Applied Computing

in the

Department of Computer and Mathematics

Declaration

All sentences or passages in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Signature:

Name:

Date:

Acknowledgments

I would like to give a special mention to all my lectures over the last three years of my study here in Waterford Institute of Technology, my course leader and a special mention to Mr. Pat McInerny who has provided me the knowledge and guidance that I deeply needed throughout the research and build phase of my final year project. I thank you all.

I would also like to thank my old and new classmates, Alex, Loti and Kacper for their valuable suggestion and comments on the various sections on my project. I couldn't have asked for better friends and colleagues to share this unforgettable experience with.

To everyone that participated in this project, specially Mr. John Claro who took his time and looked over my codes. Thank you for your time and co-operation

Lastly but certainly not the least, I sincerely acknowledge the efforts of my noble family for their constant moral support and sacrifice to give me the best possible education.

Abstract

With the rapid growth of technologies such as AI, automated data tracking or data mining in sports has become more efficient and a lot faster in extracting data, that is why it's growing high in demand. Due to major improvements in technologies data extraction applications are now being used by most sport analyst to study and output an evaluation report on the athletes capabilities. Major teams and even players themselves have acted quickly to these turn of events and are now investing in the idea of data driven decisions to win games.

This paper presents an automated task of player position and ball detection in the game of basketball that enables basketball athletes to evaluate their shooting abilities by themselves through an application built using python. The application will implement multiple technologies such as convolutional neural networks and object detection algorithm in an attempt to classify and record the difference between a make or miss shot. Both players and the ball will be detected through an input video that will then be divided into smaller images called frames. Each output frames will become an input to various inner functions to produce a scoring system for managing and counting between the make and miss shot.

This paper also presents multiple different approach in tackling one of computer visions main challenges of automated object detection by using algorithms such as YOLOv3 and using open source frameworks like OpenCV. This paper will clearly explain the process of frame extraction from a static video taken by a mobile camera, image filtering using a colour range detection method and understand the use of (SSIM) or Structural Similarity Index Measure for shot classification.

Table of Contents

1 INTRODUCTION 7

1.1 BACKGROUND 8

1.1.1 Computer Vision 8

1.1.2 OpenCV 8

1.1.3 Artificial Intelligence 8

1.2 AIMS AND OBJECTIVES 11

1.3 RESEARCH QUESTION 13

1.3.1 Classifying between a Make and Miss Shot 13

1.3.2 Image filtering for maximum detection accuracy 13

1.3.3 Pose Estimation vs YOLO for player detection 13

2 LITERATURE REVIEW 14

2.1 PROJECT SOFTWARE AND TECHNOLOGIES 14

2.1.1 MATLAB vs Python 14

2.1.2 Convolutional Neural Network 15

2.1.3 Cross Platform App with Electron.js 16

2.1.4 R-CNN Object Detection 16

2.1.5 YOLO Object Detection 16

3 METHODOLOGY 18

3.1 PARTICIPANTS 18

3.2 PROJECT MATERIALS 18

3.2.1 Camera and Input Video 18

3.2.2 Using and Storing of CNN Model 19

3.2.3 Large Input in Model 20

3.2.4 Darknet YOLOv3 20

3.2.5 HSV Colour Mapping 21

3.2.6 Image Masking and Contour Extraction 21

3.3 OPENCV VISUAL FUNCTIONS 21

3.3.1 OpenCV Text function 21

3.3.2 OpenCV Circle Function 22

3.3.3 OpenCV Rectangle Function 22

3.4 IMAGE FILTERING 23

3.4.1 Gaussian Blur 23

3.4.2 OpenCV Erode and Dilate 23

3.4.3 OpenCV Convert 24

..... 24

3.5 BUILDING THE CONVOLUTIONAL NEURAL NETWORK 25

3.5.1 Loading and formatting data set 25

3.5.2 Model Structure 25

3.5.3 Model Fitting 26

3.6 BUILD FRAMEWORK 27

3.6.1 Image Filters and Enclosing Circle 28

3.6.2 CNN Classification 29

3.6.3 SSIM Shot Classification 29

3.6.4 YOLOv3 Player Detection *30*
3.6.5 Mapping of Player Position into 2D Court Image *30*

4 RESULTS **32**

4.1 Ball Detection Result *32*
4.2 Player Detection Result *32*
4.2.1 Players mapping and Shot classification *32*

5 CONCLUSION AND DISCUSSION **34**

5.1 LIMITATIONS 34
5.2 IMPROVEMENTS AND RECOMMENDATIONS 35
5.3 CONCLUSION 35

6 APPENDICES **37**

6.1 IMAGE FILTERING 37
6.1.1 Image mask without image filters *37*
6.1.2 Image mask after image filter are applied *37*
6.2 RECORDING OF POINT COORDINATES 38
6.3 PLAYER AND BALL DETECTION 38
6.3.1 Player Detection *38*
6.3.2 Ball Detection and ball path graph *39*
6.3.3 Player and Ball Detection Combined Functionality *39*
6.3.4 Player position mapping *40*
6.3.5 SSIM Test *40*
..... *40*
6.3.6 Pose Estimator API *41*
6.3.7 Final Application *41*

1 Introduction

Over the last five years a study by Alice Bonasio concluded how advanced computer technologies can now be obtained at affordable prices and that Artificial Intelligence have constantly been growing by 12.9% annually worldwide. (Bonasio, 2018). This growth in AI led to better open source materials that can be used to build computer vision applications. Along with growing technologies, research topics on sports regarding data extraction and tactic analysis have been widely explored.

In general, basketball is all about athletes gaining competitive advantage over the opposing team and win matches by enhancing individual capabilities for better team performance. With the availability of statistical data in sports and intelligent applications instructors or coaches are able to analyse the total skill percentage of an athlete based on data from previous games. The data gathered is then used to determine what a player needs to improve on next.

Nowadays data mining in sports is a luxury that almost every athlete takes advantage on in order to evolve their level of game play. This paper is mainly focused on the different methods and technologies needed in building an intelligent tracking system to help athletes visualized their shooting ability. Using object detection with deep learning, the tracking system takes in a video input and outputs a statistical representation for every shot attempt. In short, the application calculates shot percentages using python and uses YOLO or convolutional neural networks to detect the athlete frame by frame. Every detected athlete would then be map into a two-dimensional court image to represent its position that taken directly from the three-dimensional video frame.

1.1 Background

1.1.1 Computer Vision

Computer Vision is the process of segmenting an input video into smaller frames and performing real world data extraction using various image processing techniques. The resulting information will then be use to program a computer to be able to automate tasks that human vision can achieve. It has multiple sub-domains that ranges from video tracking to object detection and motion estimation. This in turn played a major role in computer vision applications in medicine and security navigation.

1.1.2 OpenCV

OpenCV or Open Source Computer Vision was originally a small project proposed by Gary Bradsky that started at Intel in 1999. It started from being a single platform and now into a cross-platform library that supports Windows, OS X and Linux. OpenCV also supports a wide variety of programming languages such as Python, Java and C++.

OpenCV is a huge open-source library mainly intended for real-time image manipulation operations. When integrated with various python libraries OpenCV can be structured as a NumPy array of image pixels. This allows other functions to easily perform pattern matching in order to detect object features and perform mathematical operations on these features. Today OpenCV is used in applications such as face recognition, automate surveillance system and even in sport analysis.

1.1.3 Artificial Intelligence

Artificial Intelligence or AI for short is a technique that enables a machine to mimic human behaviour. AI was first introduced to us in the year 1951 where Minsky and Dean Edmunds built the very first artificial neural network using three thousand vacuum tubes in order to develop a network of forty neurons. Today AI has evolved

into a broader umbrella where the topics machine learning and deep learning fall as a subfield of AI. (Press, 2016)

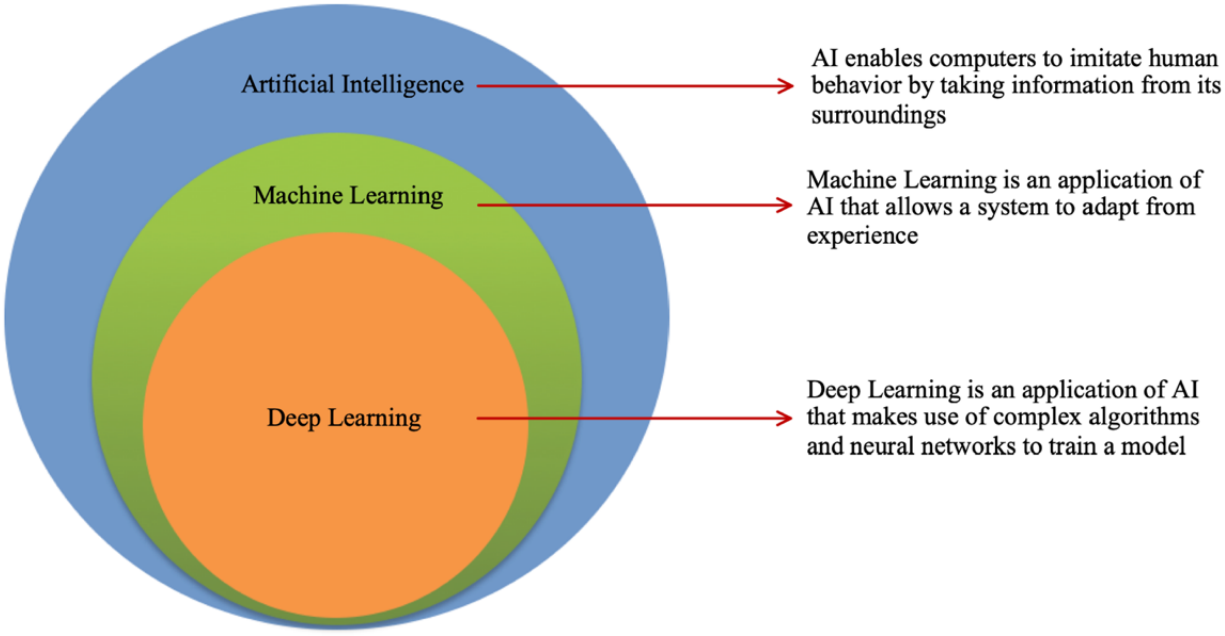


Figure 1.1: *AI and its inner applications*

Deep learning in short is an approach to achieve machine learning by training artificial neural networks using either one of the two concrete methods called supervised learning and unsupervised learning. Supervised learning trains the neural network using a labelled data, while unsupervised learning trains the neural network using unlabelled data and searches for recurring patters. This research mainly focuses on the use of convolutional neural networks but overall there are a total of four major neural network architectures: (Goodfellow, Bengio, & Courville, 2015)

- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks
- Recursive Neural Networks
- Unsupervised Pretrained Networks (UPNs)

In multiple computer vision objectives Convolutional Neural Networks acts as the basic building block for object detection due to its efficiency in classifying which class the detected object belongs to in terms of its pixel density. In context of deep learning CNN algorithm works the best mainly because in most images, objects have consistency in relative pixels which can be a major input to the convolution layers. Deep into the CNN architecture, computer vision tackles the following issues: (Gibson & Patterson, 2017)

1. Object Classification: classifying every pixel of an image and labelling them according to the class they belong to.
2. Object Localization: learning which class an object belongs to and locate its position.
3. Multiple Object Detection: detecting more than one object on a single image and classifying which class each objects belongs to.

With deep learning requiring algorithms to observe and provide accurate results of a big data set, a flexible and easy to use programming language is needed. This is where Python comes into place. Python is a flexible platform-independent language that is compatible with platforms such as Windows, LINUX and macOS. Python having many reliable open source library, this enables programmers to reduce development time and fix bugs efficiently. Overall Python offer various unique technology stacks and libraries for machine learning such as: (Gupta, 2019)

- NumPy offer n-dimensional array for data analysis and manipulation
- Pandas offer numerical tables and series for data analysis and manipulation
- Keras, TensorFlow, and Scikit-learn for machine learning
- Seaborn for visualizing data based on matplotlib

1.2 Aims and Objectives

One of the main challenges in sport apart from winning a championship is the objective of analysing and evaluating the player performance in a game as an individual. This is traditionally performed by an analyst who either watches the game live or be at the game venue in order to manually record the necessary player statistics during the game. Player statistics proved to be a new method of improving the athlete and while we have stats analysis we also physical capabilities testing which is done in the laboratory.

The aim of conducting this research is to create a build plan for creating a computer vision application that the athletes themselves can easily implement into their daily training regime. The final aim is to have a fully functional, robust Python application that is able to process an input video and fully detect both the players and the basketball. An application that will output an overview of the total shot attempts with clear labelling between the make and miss shots.

To achieve an efficient basketball detection application and to give athletes another alternative method to enhance their shooting ability, I decided that we needed to implement the following functionalities:

- extract video frames as an image from input video using Python's open source library called OpenCV.
- apply appropriate image filtering such as adding image blur to improve the accuracy of ball detection.
- using the filtered frames we need a function that locates the exact position of the athlete at compile time whenever he or she shoots the basketball.
- Display player position from three dimensional perspective onto a two dimensional basketball court image.
- display a statistical representation of the overall shot attempts, clearly distinguishing the made shots and the missed shots using coloured markers.

- create a scoring system on the frame that will be able to increment the current score at real time once a shot is detected.

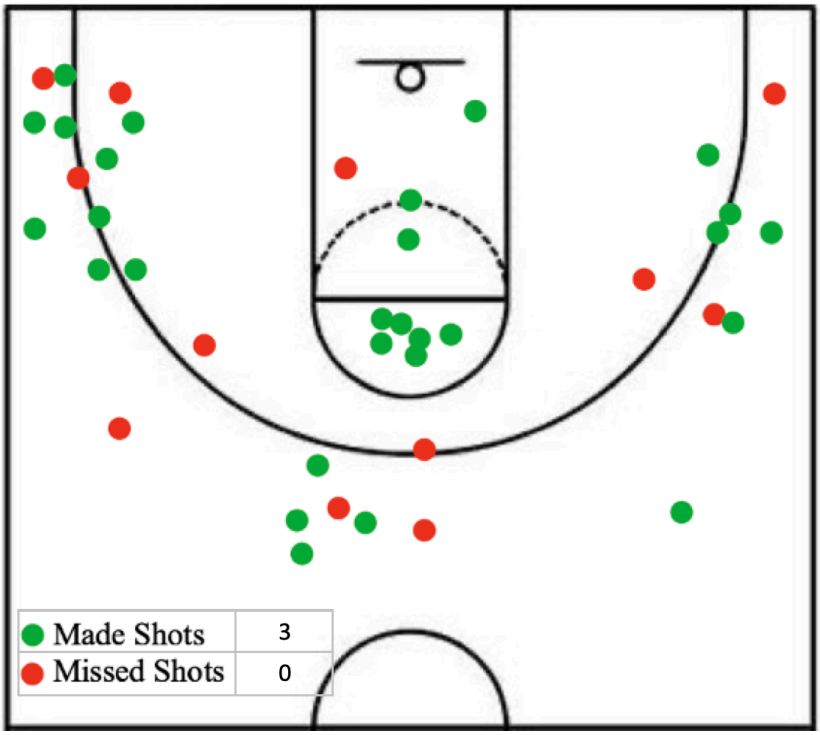


Figure 1.2: *Location of every shot attempts with score count*

Deep learning and neural networks have the reputation of providing an inaccurate probability and estimates, which results to an adversarial example. To avoid from suffering to such error, we aim dig deeper into various detection algorithms such as YOLO object detection to determined which algorithm is the most accurate. To further increase the accuracy of object detection three objectives were laid down. First objective is to investigate on how image filtering can affect either the detection of the basketball or the athlete. Second objective is to investigate about the use of OpenCV background subtraction method in an attempt to get a better person detection functionality. Investigate an API to detect player pose and see if API is runnable.

1.3 Research Question

1.3.1 Classifying between a Make and Miss Shot

To fully benefit from using the application we need to figure out ways or methods of how to determine if a basketball went in. When a player shoots the ball is there a method for finding the angle of the shot? Figuring out when to perform the classification is vital.

Also we need to point out that having a clear and see through backboard will mess up the classification if a colour detection method is used. There will be cases where the player uses too much power when shooting the ball and it goes through behind the back board. Colour detection function only detect an object based on a colour range therefore it might detect a make shot when it passes through the rim from behind. In short the ball direction and where it will bounce is unpredictable.

1.3.2 Image filtering for maximum detection accuracy

OpenCV offers various image filters that lets us change how the pixels are arrange in the image. This refracturing of image pixel values changes how it performs when passed to a detection function. However how do pick which filters to use in order to get a high detection accuracy.

1.3.3 Pose Estimation vs YOLO for player detection

There are many ready built APIs that could be useful for player detection. Is it possible to have faster compile time when using APIs in comparison to detection algorithm such as YOLO, and is Pose Estimation a better alternative for player (x,y) coordinate extraction?

2 Literature Review

In this section, the literature on the selection of software, technologies suitable for the application and the constraints of using either a phone camera or a digital camera will be explained.

2.1 Project Software and Technologies

2.1.1 MATLAB vs Python

Over the years computing has grown into a much bigger community with people continuously evolving the use of languages such as Python and MATLAB. Nowadays when it comes to analytical and quantitative computing Python is at the top compared to MATLAB mainly because of its unique libraries, straightforward syntax and a huge community support. MATLAB is an array-based programming language that also offers similar scientific libraries similar to Python. Libraries include data visualization and image processing which is similar to matplotlib and OpenCV in Python.

In deep learning the execution time for an algorithm is a crucial factor specially in situations where execution is real time. Based on the experience of working with both languages I have decided to go forward with the project using Python. Main reason is that MATLAB is not great when implemented into a web application and a license need to be purchase for every MATLAB toolbox before use. Python on the other hand offers a free web application framework called Django or Electron and all its library can be installed for free on the command prompt.

Based on the overall outcome and finding from semester one it was concluded how Python is much more flexible in terms of open source computing libraries for computer vision modules compared to MATLAB. Also Python is mainly a scripting language so its most compatible with OpenCV. While Python excels as a scripting

language , MATLAB on the other hand is more compatible with forensic image processing types of projects.

2.1.2 Convolutional Neural Network

The study of machines having the ability to have the same vision as us human is getting more realistic thanks to the concept of neural networks. Concepts such as automated self-driving cars to facial recognition has been made possible with the use of cutting-edge image recognition technology called CNN or Convolutional Neural Networks.

Convolutional neural networks operates by exploiting different properties from the image. Feature exploitation in image is possible mainly because of how images are structured like how image pixel values are stored in a two-dimensional array. CNN examines each pixels values in an attempt to discover potential patterns. Image pixel have values in them to represent a colour value, in turn these values are in the form of a three value tuple. Tuples in the form of RGB (red, green and blue) values, respectively each values ranging from 0-255 (En, 2019).

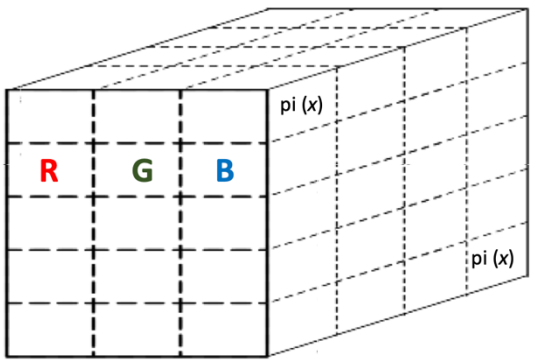


Figure 2.1: *Image Pixel Format*

When video frames are processed as an input to a CNN model, the model extracts image dimension values in order to calculate the amount of features to be exploited. The formula to calculate the length of all features is (width * height * 3). Usually with RGB images the feature length reaches almost five hundred thousand, it is such a big amount for feature exploitation therefore it might take quite a long time to compile. This is usually fixed by resizing the frame to a much smaller values and to transform the image from an RBG (three dimension) format into greyscale (two dimension) format. Grey contains one less dimension given us a feature length that varies from six thousand to forty thousand.

2.1.3 Cross Platform App with Electron.js

Atom or Electron.js as we know now is an open-source framework developed and maintained by GitHub. Electron.js allows the build of creative cross-platform applications in the web using basic technologies like HTML, CSS and JavaScript. Building an interactive interface in Electron for your Python application is made easier with the help of a GUI toolkit that allows us to create attractive buttons and menu.

2.1.4 R-CNN Object Detection

R-CNN or Region CNN is another method for object detection in deep learning. This method can be summarized into three steps. First different image proposal is created for a single image. Secondly for every proposed image an SVM classifier is used to detect what class the object on the image belongs to. Lastly every classified image is overlapped and the proposal with the highest score it picked. R-CNN is currently the second fastest and most accurate detection method where it once scored 54% on PASCAL VOG 2010. (Ananth, 2019)

2.1.5 YOLO Object Detection

You only look once algorithm or YOLO for short is currently the fastest object detection algorithm today, YOLO is a one-stage detector strategy that only looks at the input image once before detecting any classes on the image. It does this by separating the image into an S by S grid where each grid cell is responsible for predicting five bounding boxes. Bounding box in short is a description of the rectangle that encloses the object in the image. Lastly YOLO outputs a confidence score telling us how certain the system is that the predicted bounding box actually encloses an object. (Sharma, 2018)

YOLO has become so much faster and better after a number of iterations. It started from being to detect two object, but after a joint training using the dataset of ImageNet and the COCO dataset it can now detect over nine thousand objects.

Recently a newer version of YOLO called YOLOv3 has been introduced by Darknet. Darknet has optimized the algorithm and picked the top relevant objects that the model will detect. Darknet reduced the detectable objects to an estimated of ninety labels. Darknet also implemented a concept called NMS or non-maximal suppression. After the adding NMS onto the YOLOv3 the algorithm has transformed into a much more efficient object detection by only keeping the best bounding box or best confidence scored. The increase in efficiency have its trade-off, while the compile efficiency increased the compile time decreased drastically.

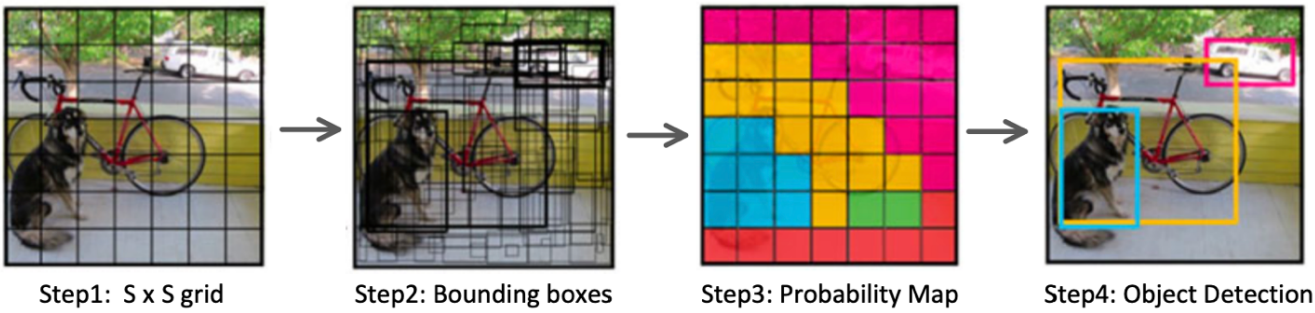


Figure 2.1: *YOLO Object*

3 Methodology

In this chapter, the methods in order to acquire the data to be used in the project will be discussed. The people who participated on getting images, the various materials to be used in building the application and the outline of the build plan will all be discussed.

3.1 Participants

When building a convolutional neural network a large dataset is needed to train the model. There are thousands of basketball and basketball hoop images in the web but I would've have to reference each one, therefore to solve this issue I did the following:

- I posted a 'yes' or 'no' vote in my Instagram and snapchat story asking if anyone with a basketball is willing to help me for as little to less than five minutes.
- after a day from posting the story, I checked the voting result and privately messaged everyone that voted yes.
- I sent a message to ask them to take a five photos of their basketball and send it me.

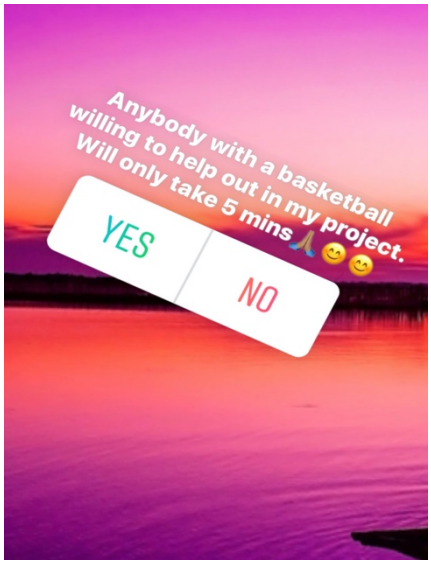


Figure 3.1: *Instagram Story Poll*

3.2 Project Materials

3.2.1 Camera and Input Video

The initial goal for the final project was to build a mobile application that is able to perform real time execution using a mobile camera as the source for the input data. After the research phase at semester one and brief study on the given time frame for semester two I've decided to change the overall concept. Instead of building a mobile application I'm planning to build a web application using the electron framework.

Web application execution will not be performed on real time and will take an input video locally from your machine instead.

Having removed the mobile application in the main build plan, it is now important to provide the application with the best resolution input video. Nowadays most mobile cameras would've set us up with a decent quality data input for our detection system. The main rule to follow when finalizing the input video is if the basketball and athlete can't be recognized by you, then there's a high probability the system can't detect any objects.

3.2.2 Using and Storing of CNN Model

Convolutional Neural Networks is such a big architecture and the building a new model from scratch each time would be time consuming. That is why the implementation of persistence in our application saves us processing time by re-using the trained model rather than training a new model. JSON which is the standard file exchange format will be used in both prototype and final application. NoSQL a cheap non-relational database that follows no structure is chosen as the database for the application because of its ability to scale with increasing data. This enables us to train a pre-trained model efficiently and the use of JSON as the main storing format gives us easy access to our data set. The easiest way for storing and re-using our model is to use TensorFlow and Keras to save the model locally in our machine. The following code for saving and accessing the model is shown below.

```
# export our model so we can use it for other projects
model.save('BasketballPlayerDetectection/MODEL')

# load our model
model = tf.keras.models.load_model('BasketballPlayerDetectection/MODEL')
```

Figure 1.5: *Saving and Loading CNN model*

3.2.3 Large Input in Model

An alternative method to feed large images in RGB format to a CNN model is by dividing the frame itself into four quadrants frames. The model would output a detection score on all four quadrants. The quadrant with the biggest score will repeat the same steps until the ball is finally detected. This method is very time consuming because the algorithm performs multiple iterations. The following methods follows the similar concept of the original convolution, the only difference is that it has four frames to run on.

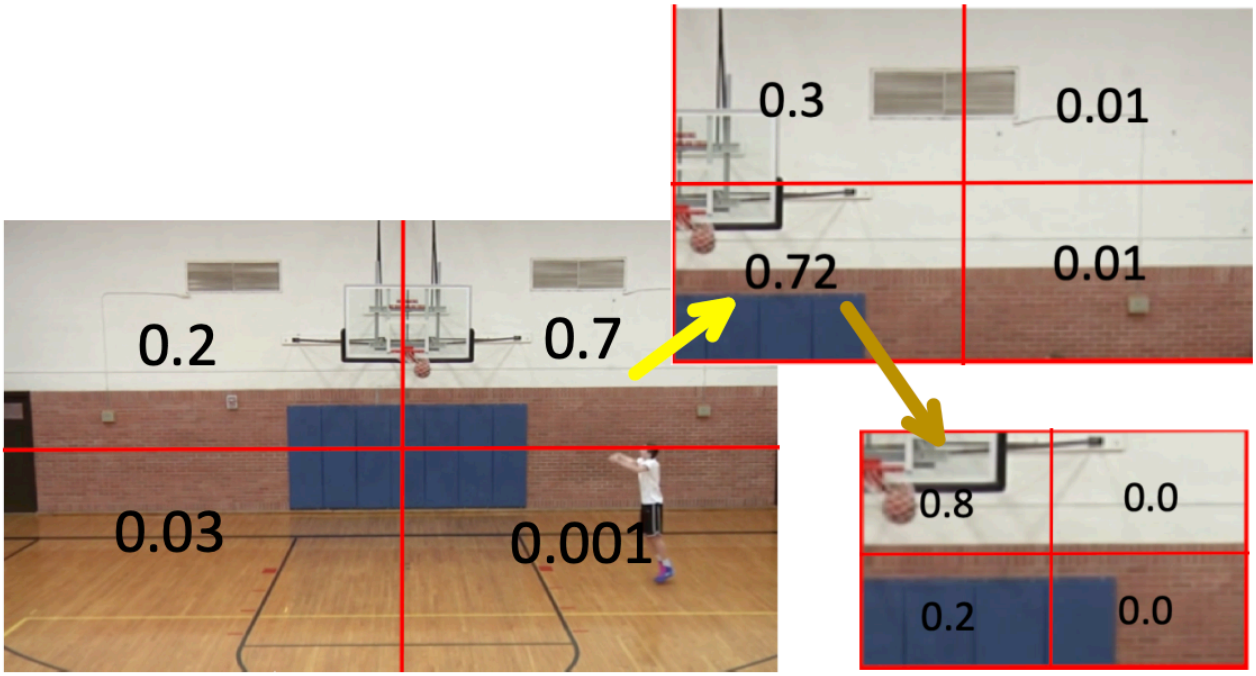


Figure 3.2: *CNN method for dealing with large images*

3.2.4 Darknet YOLOv3

YOLOv3 is the newest version of the YOLO object detection algorithm. YOLOv3 differs from rest because it's refactored to detect objects with the highest accuracy. There is a massive benefit to the increase of detection accuracy, however a trade-off had to be made to make this possible. The trade-off led to the decrease of the algorithms compile speed running at an average of 30 FPS, this is mainly due to the addition of an complex architecture called Darknet. (Kathuria, 2018)

3.2.5 HSV Colour Mapping

HSV or Hue, Saturation and Value is a different approach to represent a colour and a lot easier in terms of object detection based on colour compared to RGB colour - space. Normally colour detection algorithm iterates through every pixel in the image and searches for a specific colour, but with HSV the algorithm search in terms of colour purity and colour position.

The RGB colour model uses three main colours (red, green and blue) each having a value that ranges from 0 to 255. RGB forms its colours by adding different values of red, green and blue. The HSV colour model uses three dimension, hue represents the colour, saturation represents the dominance of that colour and value represents the brightness of the colour.

3.2.6 Image Masking and Contour Extraction

Using the calculated lower and upper range of selected colour a mask is create to be compared to the main frame. Masking is the process of keeping any pixel that is in the HSV colour range and discarding every pixel outside the colour range. Once the mask is applied we end with black and white representation of the frame, where everything that's colour black isn't in the colour range we specified and everything is white is objects that passes the colour range. A OpenCV function that runs through all the pixels in the image. This function in turn will output an array containing all the pixel (x, y) points of the image that's colour white.

3.3 OpenCV Visual Functions

3.3.1 OpenCV Text function

In our application the process labelling objects or outputting a message to the window is highly necessary in order to keep the user intact of what is happening. OpenCV offers a built-in function that enables us to a fixed display text or display

text depending if the passed conditions that we created is met. The OpenCV put text function takes in a total of seven inputs and the inputs are as follows:

Input 1: the frame or image that the text will appear on.

Input 2: the text to be displayed.

Input 3: the (x, y) coordinate to where the text will be position on the frame.

Input 4: OpenCV text font

Input 5: text font size

Input 6: the font colour value in the form of RGB

Input 7: the weight of the text, default value of 1

3.3.2 OpenCV Circle Function

To clearly visualize our detected basketball and to showcase the detection function working the OpenCV circle function will be in use. The OpenCV circle function lets us draw an circle given the centre point, the radius and the frame that the function will draw the circle on. In total the OpenCV function takes in a total of five input parameters and they are the following:

Input 1: the frame or image that the text will appear on.

Input 2: the (x, y) coordinates or centre point of the circle

Input 3: the radius of the circle in integer format

Input 4: the colour value of the stroke in the form of RGB

Input 5: the thickness of the stroke

3.3.3 OpenCV Rectangle Function

The rectangle function is very similar to the circle function above the only difference is instead of the radius we have a value for the width and height of the rectangle. This width and height value can be dynamic but It can also be fixed depending on your preference.

3.4 Image Filtering

Image filtering is an important factor in our application, it is used in every input frame taken from the video. Filter are used in order to increase or maybe decrease the image quality in order to get the best possible objection detection accuracy. This section explains the overview of every filters used in the project.

3.4.1 Gaussian Blur

Gaussian blur is the action of multiplying pixel values with a low-pass gaussian kernel filter. This is very useful for removing noise from image as shown in the figure 3.3. It does this by removing or changing the values of pixels with a high frequency. In the input parameters of the gaussian function the width and height should be specified as a positive and odd value.

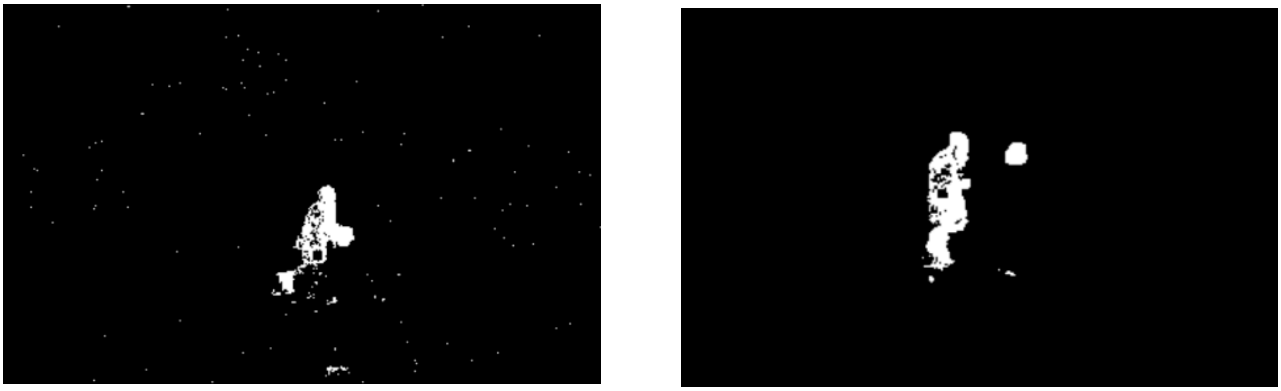


Figure 3.3: *Before (left) and after (right) Gaussian Blur Filter*

3.4.2 OpenCV Erode and Dilate

In image processing the two most comon morphological functions are erosion and dilation, both have a wide array of uses from noise removal, isolating an individual object etc. Both function are similar in fact erosion is sometimes referred to as the sister of dilate. Dilating the image is the scanning of a kernel B over the image and then compute the maximal pixel value that overlapped B and replace main image

with that of the maximal value. The Erosion is the exact same concept but instead of computing the maximal value we compute the minimal value.

Figure 3.4: *Use of Dilation and Erode Image Filters*

3.4.3 OpenCV Convert

When image is read by OpenCV it uses the default image format of RGB which is a three dimensional array of pixel however some of our functions require the image to be a two-dimensional image. That is where the convert function comes in, the function simply converts the selected image from one colour space to another. The channel of conversion always have to be specified as one of the input parameters.

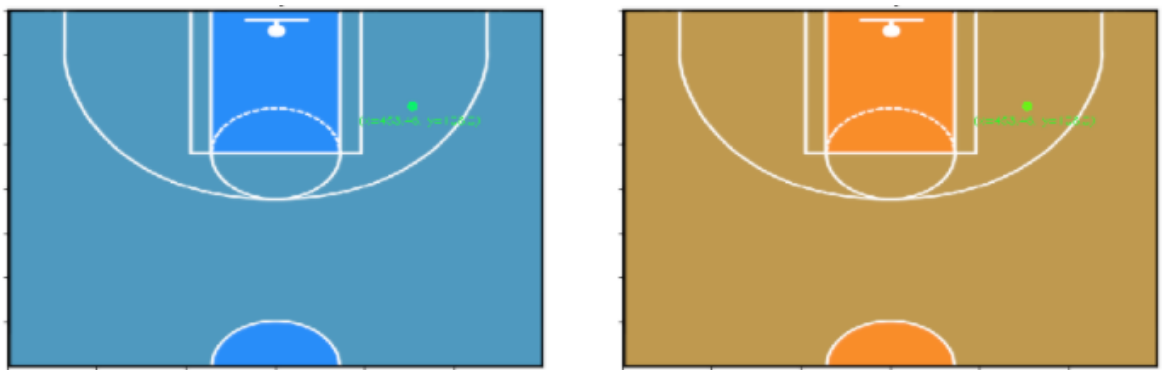


Figure 3.5: *Use of the OpenCV convert to transform HSV (left) to RGB (right)*

3.5 Building the Convolutional Neural Network

As part of the initial project objectives the build process of a convolutional neural network for classifying between a basketball and basketball hoop will be discussed.

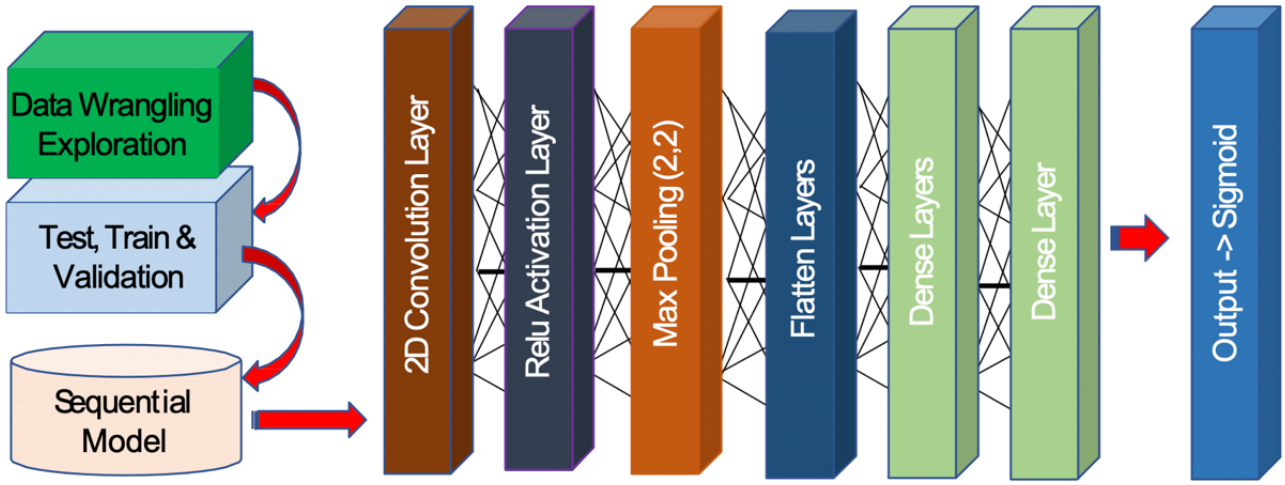


Figure 3.6: *CNN Build Structure*

3.5.1 Loading and formatting data set

Before a CNN model can be built it needs data to train and test on, so I got the basketball dataset from friends and got the hoop dataset from Kaggle. The data that will be used to train the model have to refactored first and we need to get an overview of what we want the model to be structured as. Before training the model all the images needs to be change from an RGB(three- dimensional) format into grayscale (two-dimensional) format and the image size also to be change to 80 by 80. Once all images are correctly structured every image is appended along with its label of one or zero into a NumPy array.

3.5.2 Model Structure

The model to be build is a linearly stacked two-dimensional model called 'Sequential' that will output a value of one (basketball) or zero (hoop). Keras and TensorFlow provides us with flexible methods of defining our model but in this

project were classifying between two object therefore a two-dimensional convolution with and input size two is enough. Then we add max-pooling layers to the model and also add an activation layer with a function called 'relu' which outputs the label if positive and outputs zero if not. The model layers is then flattened or compressed using the Keras 'flatten' function and lastly a logistic function called sigmoid is added so we only get an output of 0 or 1.

3.5.3 Model Fitting

Once every layer has been tuned using a method called hyperparameter tuning, we will compile the model with a binary loss function and an optimizer called 'adam'. Once compiled we fit the model eight times using the dataset that we refactored and a validation split of 20%. Then we simply save the model so it can be loaded and use by any application. The model is only trained with an epoch value of eight because looking at the accuracy graph generated by TensorFlow we can conclude how its peaking and reaching 1 at epoch value of seven. Generally the best epoch value to pick is the value before the model reached an accuracy of 1, doing this prevents the over fitting error,

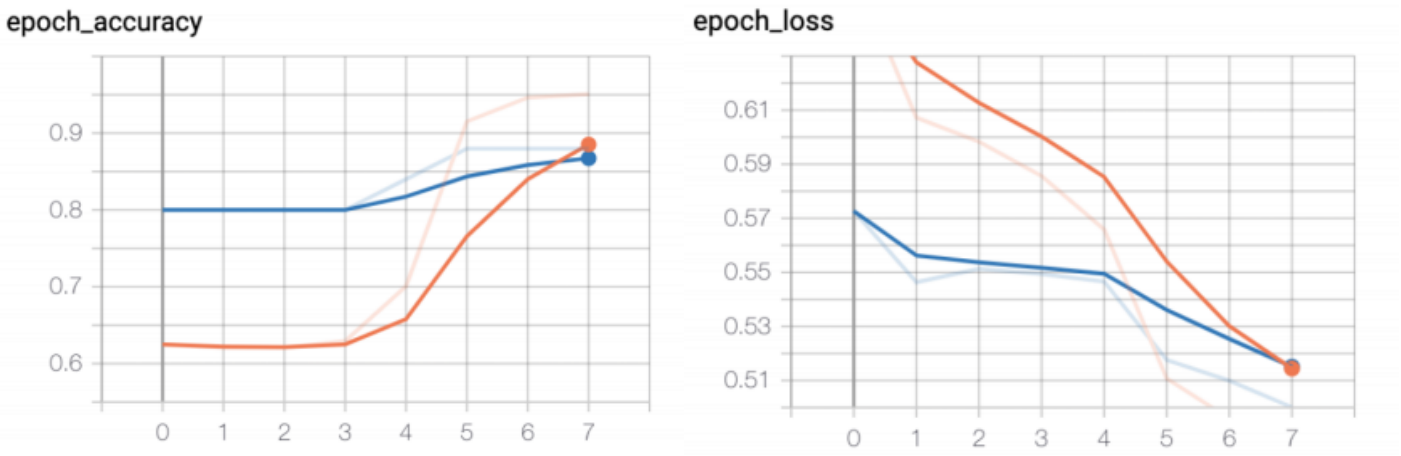


Figure 3.7.2: *Epoch Accuracy and Loss result (graph)*

3.6 Build Framework

In this section the main build framework that's was followed to achieve the final application will be describe in great detail. The build frame work can be mainly divided into three sections.

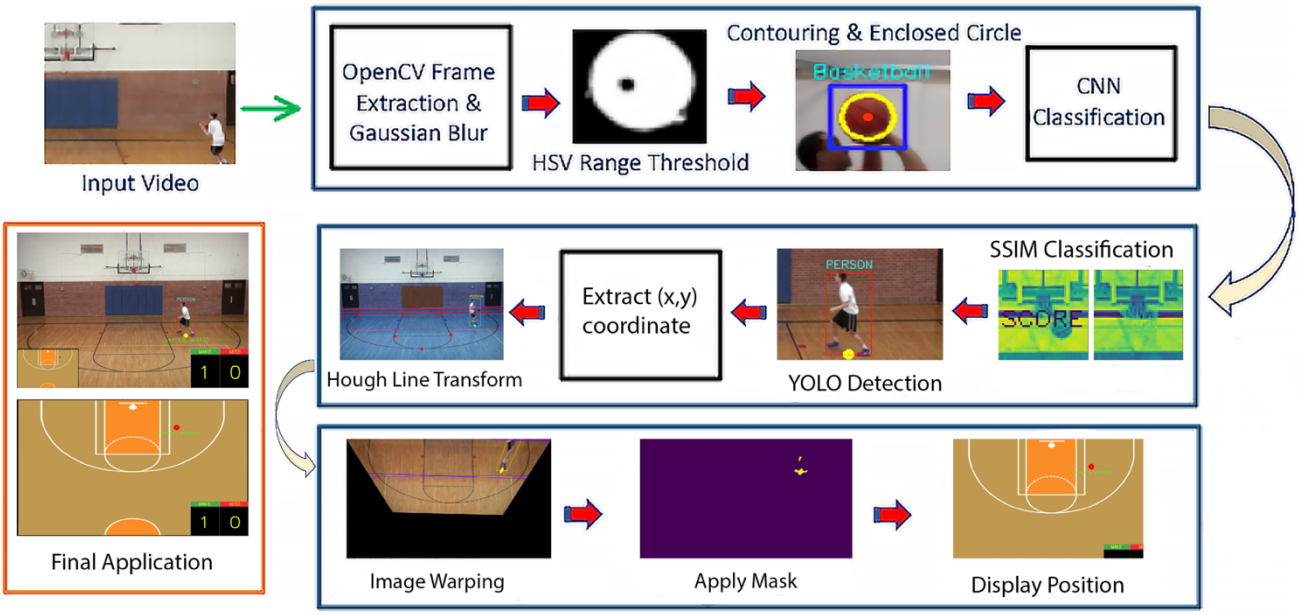


Figure 3.7: *Project Build Framework*

- In section one the main objective to perform the necessary image processing in order to detect the ball and then the detected ball is extracted as a copy from the frame to be passed on the classification model.

- In section two the objective is to classify between a make or a miss shot, using the YOLO object detection algorithm detect the player and extract its poising in the format of (x-coordinate, y-coordinate).

- Then lastly on section three the mapping of the three-dimensional points to a two-dimensional image of the basketball is performed. This in turn enables the application to visualize the position of the player at real compile time.

3.6.1 Image Filters and Enclosing Circle

To start off we get an input video that is locally stored in my machine, then the following input video is compiled using the OpenCV video capture function. The video capture function splits the video into frames, once frames are extracted a loop is initiated and will only stop until there are no frames to be extracted from the video. All inner and outer the functions in the project are then compiled inside the loop so the we won't have any functions compiling when the video has reach the end or maybe the video has not started yet.

First OpenCV image filters is applied to our frame to increases the accuracy of detection. Filters mainly reduces the amount of pixel values to exploit in the image by removing unwanted pixel noise. The first filter applied to the frame is called Gaussian blur, blurring the image removes the unwanted noise as shown in figure 3.3 which results to less invalid contours. The eroded filter then removes all the smaller objects and we use dilate to return the size of the remaining contours. The dilated frame is transformed into a HSV format and passed onto the 'inRange' function from OpenCV to create the mask. This mask will then be used to detect the contours which is in the format of (x, y). We perform an iteration through all detected contours and apply an enclosing circle function. The enclosing circle function creates a circle with the contour value as its centre, then with an 'if-statement' we are able to filters circles with radius below five.

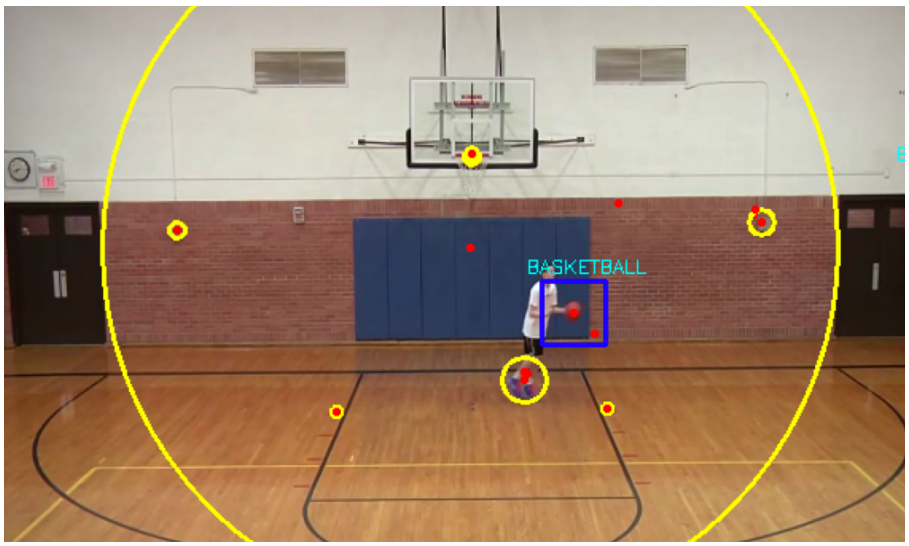


Figure 3.8: Enclosed *circle to all detected contours after image processing*

3.6.2 CNN Classification

To perform image classification we first need to extract a valid frame, this is done by creating an outbound box around the circle. The box is calculate by adding and subtracting five to the radius, with the circle centre as the starting (x, y) coordinate for the box. To extract a copy of the ROI or region of interest that we require, we simply use the same values from the bounding rectangle. The ROI will then be classified in our model to see if it's a basketball or not.

```
cv2.rectangle(frame,
              (int(x-(radius-5)), int(y-(radius-5))),
              (int(x+(radius+5)), int(y +(radius+5))),
              (255,0,0),2)

    detected_object = detected_object[
        int(y-(radius-5)):int(y+(radius+5)),
        int(x-(radius-5)):int(x+(radius+5))
    ]
```

Figure 3.9: Code for outbound rectangle and code to extract area inside the box

Now the extraction of our ROI is complete we can feed the image to a function that prepares the image before prediction by refactoring the image size and image format (GRB2GRAY). Then we used the 'model.predict' method to predict if the image is a ball or not, then using the labels we can map 1's as hoop and 0's as basketball.

3.6.3 SSIM Shot Classification

SSIM or Structural Similarity Index is the calculated perceptual metric that is usually used to define the quality of digital cinematic images and videos. It is used in the project as a method of telling the application whether we get a miss or make shot. The initial concept was to start a function that detect a miss or make shot when we detect a player shooting the basketball, but the player shot position functionality was not reached therefore SSIM is used as its simple and it is not too heavy of a load to the GPU. The SSIM function gets two images at its input parameter and compares both image to each giving a similarity value between 0 to 1, where a value of 1 means

both images are identical and a value of 0 meaning no similarity between the two images.

The output value is then used in function by using a simple if-statement that increments current score value by and display a score text on the frame whenever we an SSIM value that is greater than 0.9. The current made and miss counter will be displayed on score board.

Figure 3.10: Score Board

3.6.4 YOLOv3 Player Detection

You Only Look Once is a real-time objection algorithm that is manage by Darknet. YOLOv3 is used in the project in an attempt to detect the player in the frame and use the point of detection to obtain the player's coordinate on a 3D plane. YOLOv3 has been trained to detect over ninety daily life object such as laptop and a car, but the only object desired in the application is either a person or a basketball. Using the YOLO algorithm I have created a function that takes in an input label. The input label is then use in an if-statement where I compared the input label to all the items in the detected object array. If the input label and detected label are the same we perform visualisation otherwise do nothing.

3.6.5 Mapping of Player Position into 2D Court Image

The main objective of 3D to 2D conversion is to pinpoint the location of the athlete on the basketball court. The first stage is to use both the Canny Edge and Hough Transform algorithm in order to detect the hard lines on the basketball court. To

obtain the (x, y) points on the 3D court a function that calculates the point of intersection between two lines is used on every detected lines from the Hough Transform and Canny Edge algorithm.

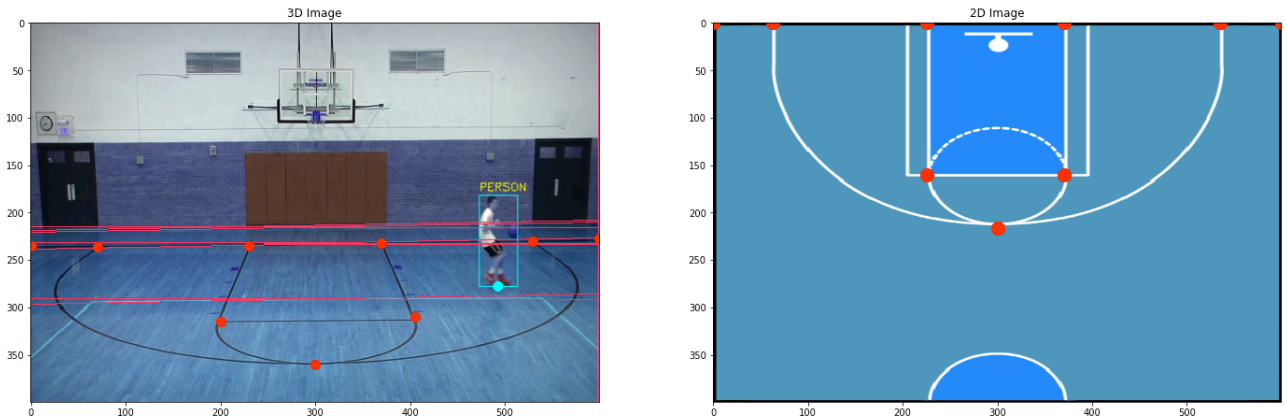


Figure 3.11: 3D (x, y) points and corresponding 2D points

Using a homograph and a perspective warp function we can warp all 3D court (x, y) points to the corresponding known (x, y) coordinate in the 2D court. Next a mask is applied using the HSV colour range of the colour yellow. With the mask applied we use the contour extract function to obtain player points and used the (x, y) coordinates of the contour with the biggest area as the of the player coordinate. To finish things off the obtained coordinates is plotted on the 2D court.

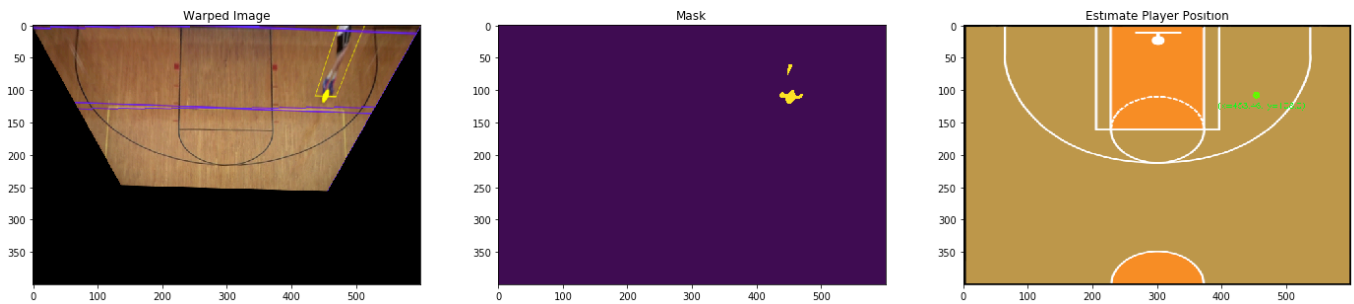


Figure 3.12: Player position extraction

4 Results

4.1 Ball Detection Result

During testing for ball detection when passed with a high resolution video the OpenCV works the best in comparison the YOLO ball detection however when the input resolution is bad YOLO out performs OpenCV. The main difference during the test is the time the function terminates, OpenCV finished the faster with 70.3 elapsed time while YOLO finished with 269.08 elapsed time. Overall the application can detect and record a basketball.

4.2 Player Detection Result

YOLO player detection passed the test and was able to perform player detection efficiently. Compared to the ball detection test where in some frames no detection occurred YOLO detection detected with an efficiency rate of 96.3%. However the time it finished or the compile time for player detection is super slow, this confirmed the discussed trade-off when YOLOv3 was being built.

4.2.1 Players mapping and Shot classification

The ability of the application to identify and track player position extraction for 2D court mapping is impressive, it worked perfectly as planed and there was very little to no errors. The main reason why the 2D mapping worked so well is thanks to the efficiency of the YOLO algorithm. The mapping function depends on the result from the YOLO detection and since YOLOv3 performs at high efficiency and accuracy the 2D mapping really benefited from it. The only down fall on the application is the scoring system. The increment function works every time a value of 0.9 is detected, but since the overall runtime is so slow it calculates three extra times or each frame thus messing up the increment system.

Figure 4.1: Final Result

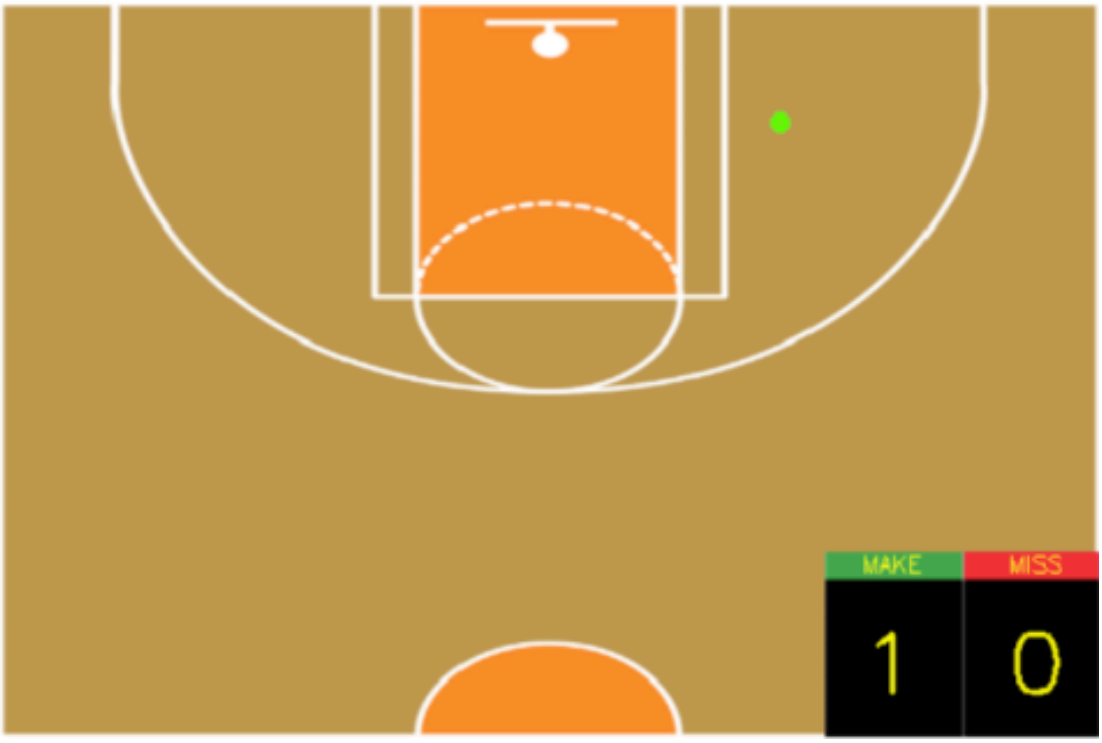


Figure 4.2: 2D court result

5 Conclusion and Discussion

5.1 Limitations

The duration of given time for the development phase is only eight weeks and with so many aimed functionalities I had to cancel out the automated (x, y) points detection. Instead of an automated points detection I hard coded the 3D points to showcase and proved the proposed concept works. Also gathering of the data set alone required around one week and it took me another two to three weeks learning on the steps on the building of a detection model using TensorFlow and Keras. After completion of application testing and debugging had to be performed, this alone took nearly one week. In short the time frame was quite short for me and an extra two to three weeks would made such a big difference.

Another limitation pointed out is that with MATLAB a licensed needs to be purchased before the software is ready to use. This did open up a better option which was to use Python instead on the initial software and programming language picked in the project proposal.

The given time frame limits the idea of building a mobile application, which would've been more reasonable for athletes as they can use the application anywhere when they train. Mobile application tends to be more successful compared to a web application mainly because people prefers quick and easy access to application nowadays.

There is also a limitation to the computational power for me to use. Object detection applications requires a lot of GPU power and will be time consuming if compile in small laptop. It is preferable that the application was run on a PC to visually see the app fully. Due to corona virus I was limited to online sources only and it prevented me from taking my own video to processed.

5.2 Improvements and Recommendations

The main improvement to apply in future would be to used my very own video file for testing and also to have a longer period of two to three weeks to allow me to work on the concept of automated 3D points detection. One main improvement that can implemented onto the application is to add a better overview of their shooting ability by using graphs. This can be done by adding a calendar function on the application that automatically set the current date on the app and whenever the app is used the data of shot attempts is stored along with the date. A weekly graph of the days versus the percentage can then be produced.

Coloured heatmaps can also be implemented on the application. This feature adds a colour label on the location of shot attempts on a two dimensional image of the court. The colour is then based on the overall percentage shots taken by far and can range from green being average and reg being hot.

Improvement for the application in the future would be create an android application version of it. Along with the android version another improvement would be to improve functions in the application so it is able to process live input video coming from the mobile camera. This overall improvement would satisfy user need of portable training assistance, an application that works anywhere and anytime as long as a valid input video is present.

5.3 Conclusion

Upon completion of this paper we have presented a working framework for automatic player and ball detection. This application can make training much more fun and informative for basketball athletes. This paper have accurately concluded that complex object detection can be made with simple technologies such as OpenCV and also the concluded that we don’t always need the newest or strongest technology to have a working semi application. The rate of or speed of compile cant deeply depend

on the algorithm used and we can say that YOLOv3 is indeed so much more accurate but a lot slower compared to others algorithms.

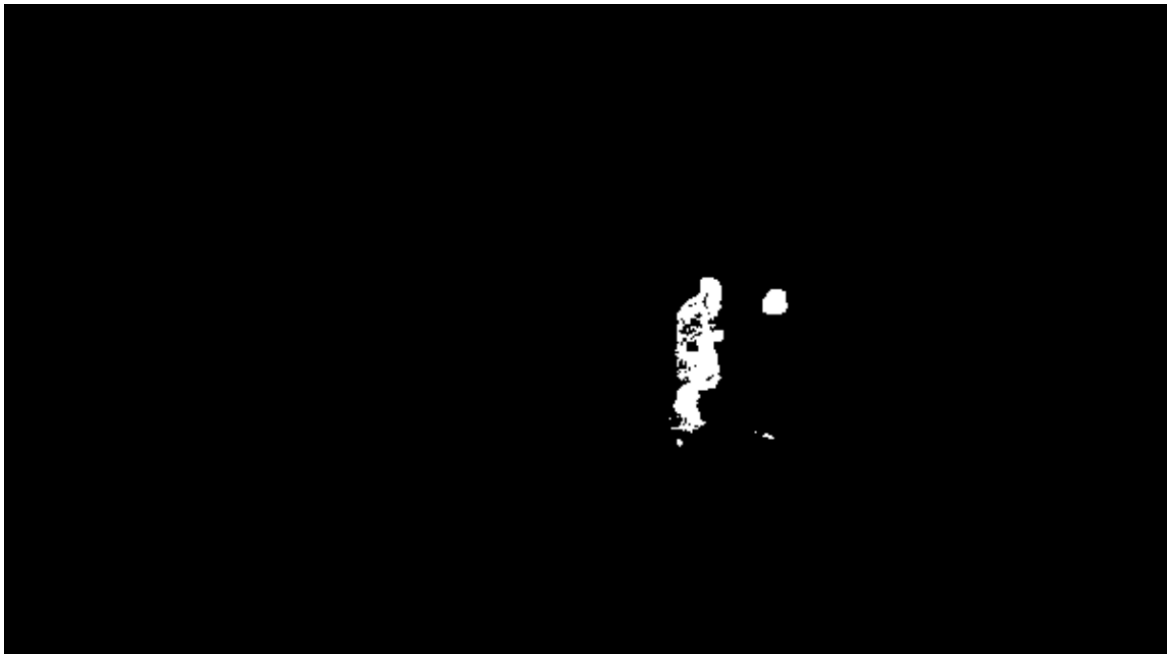
This paper showed that deep learning is a subfield of a bigger architecture called machine learning and that the concept of computer vision can be tackled with deep learning techniques. Using simple image classification and localization objects can be detected on an image. Writing neural network algorithms to train a model is the process of training the model to recognized a group of objects or classes present on an image. During the process of detection a probability score is used to determine which class the object belongs to, then a box is inserted around the object along with its class name to classify that it is detected. This paper concluded that using Python is more reasonable compared to MATLAB because of its first it is open source, second is its flexibility as a programming language and lastly Python has a massive community for support. Python also offers multiple advance libraries which can be implemented onto our project to lessen the line of code and improve execution time.

6 Appendices

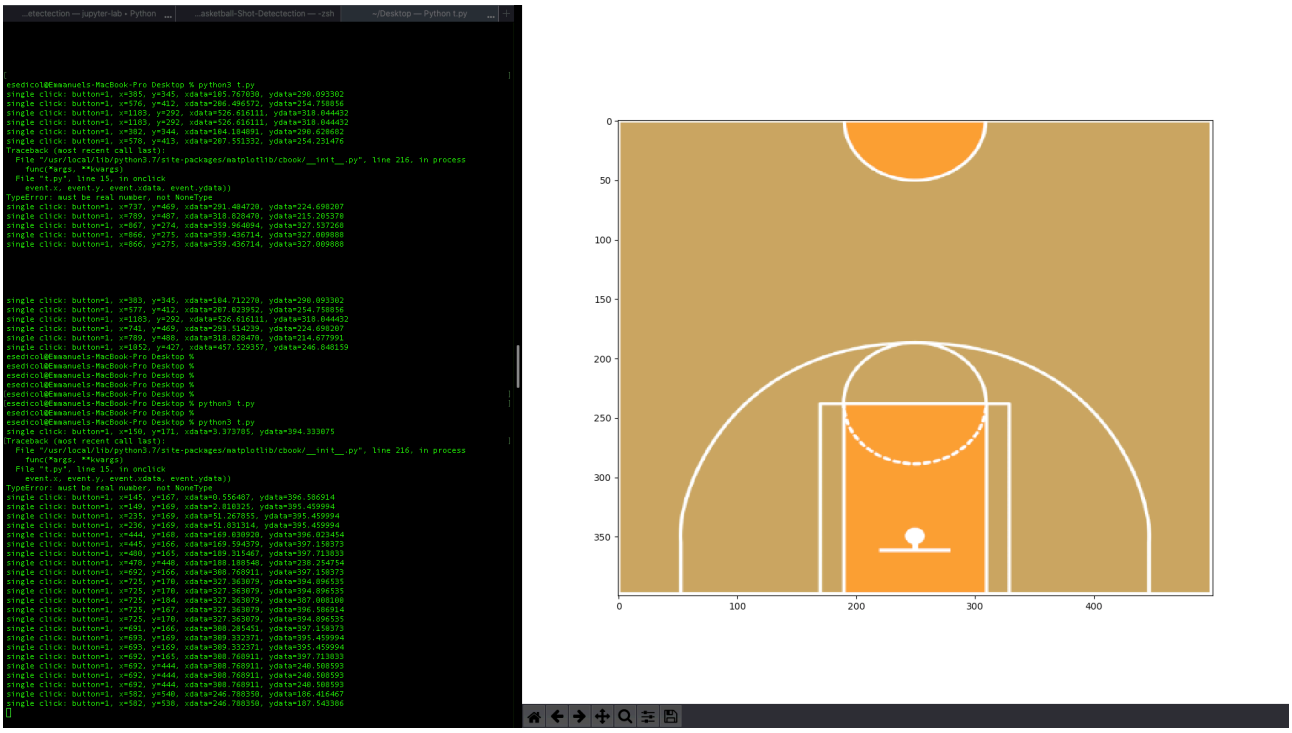
6.1 Image Filtering

6.1.1 Image mask without image filters

6.1.2 Image mask after image filter are applied

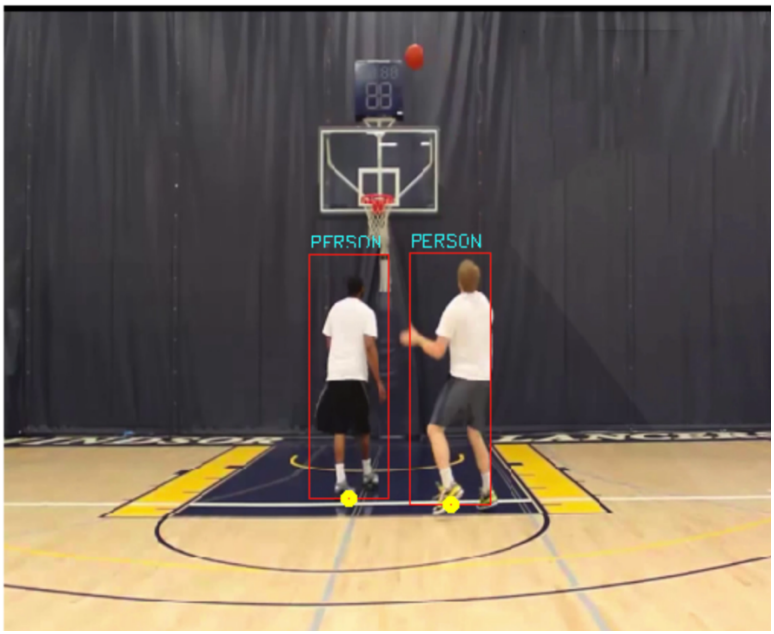


6.2 Recording of point coordinates

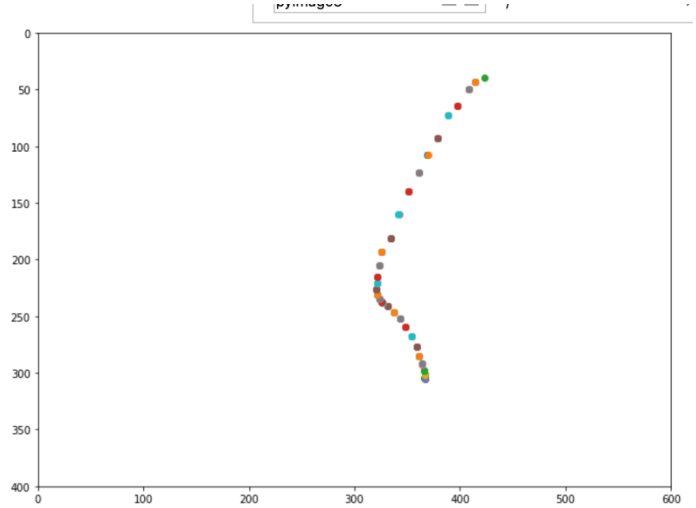


6.3 Player and Ball Detection

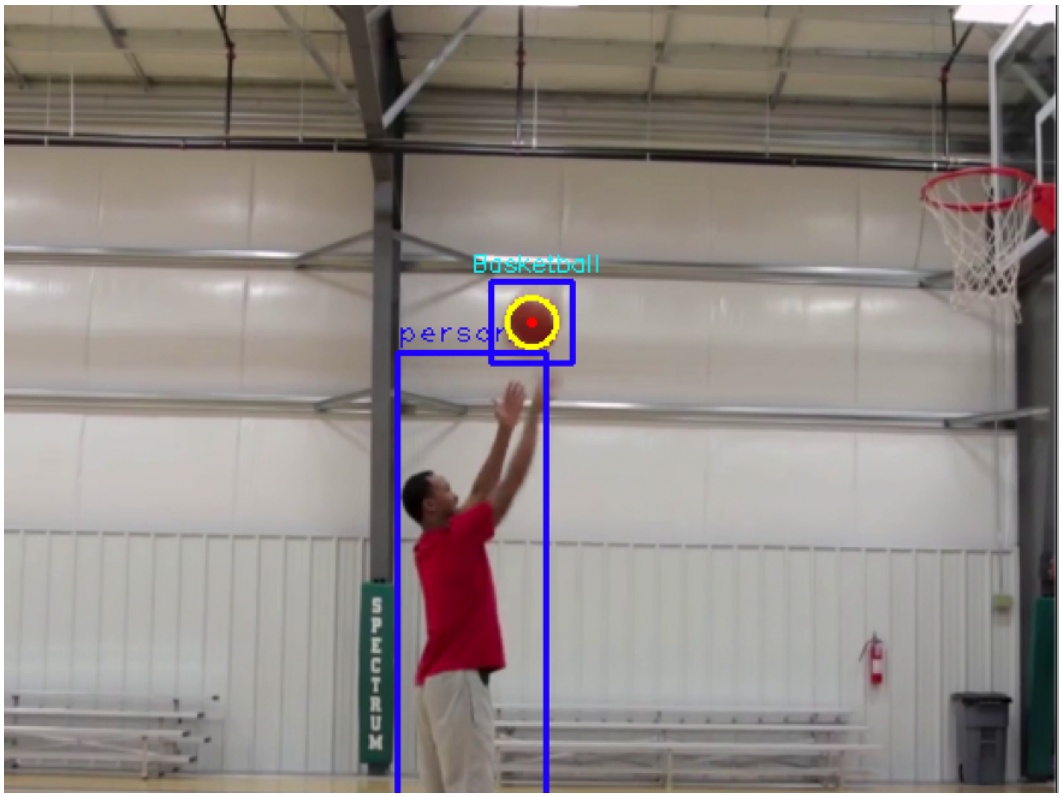
6.3.1 Player Detection



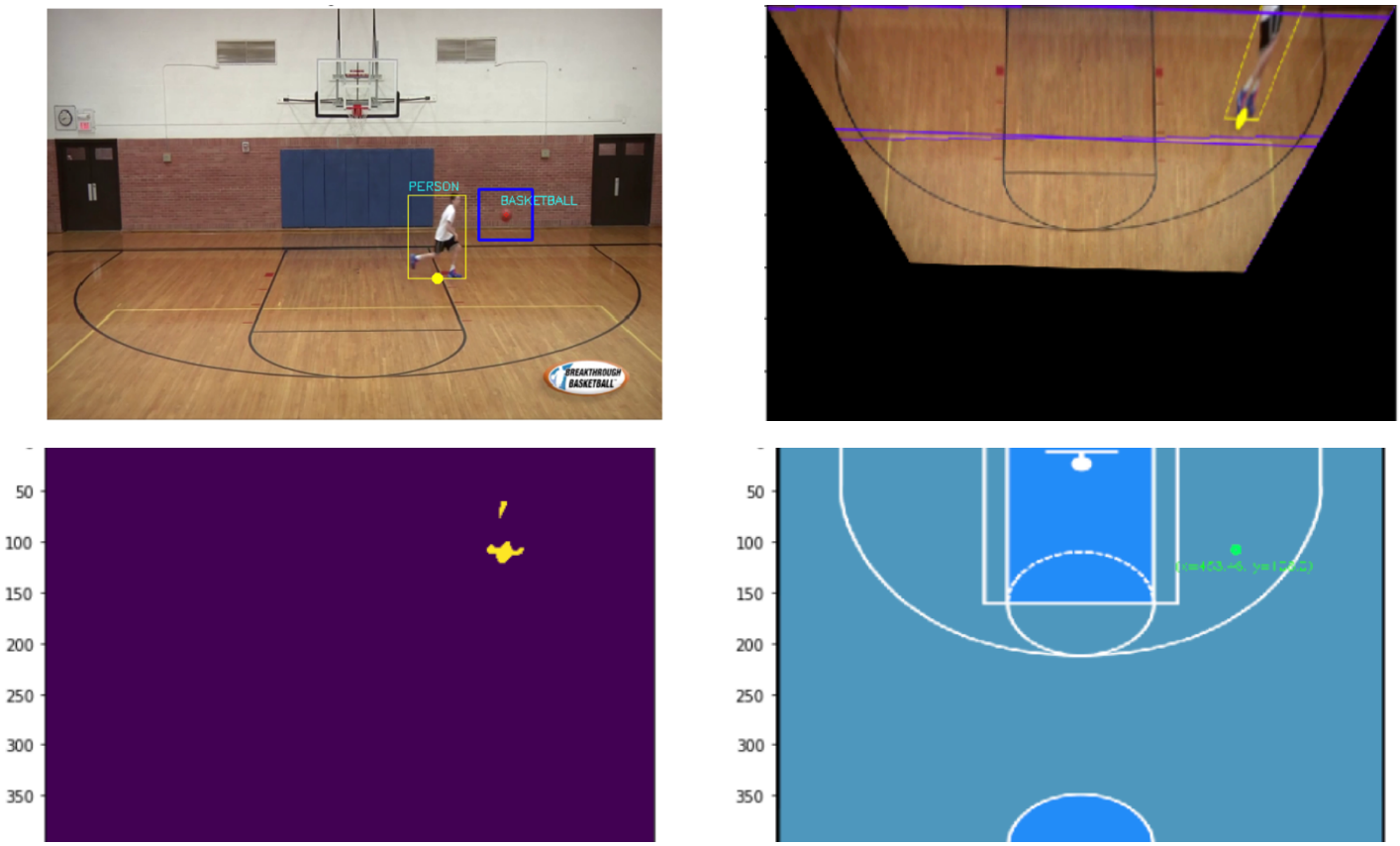
6.3.2 Ball Detection and ball path graph



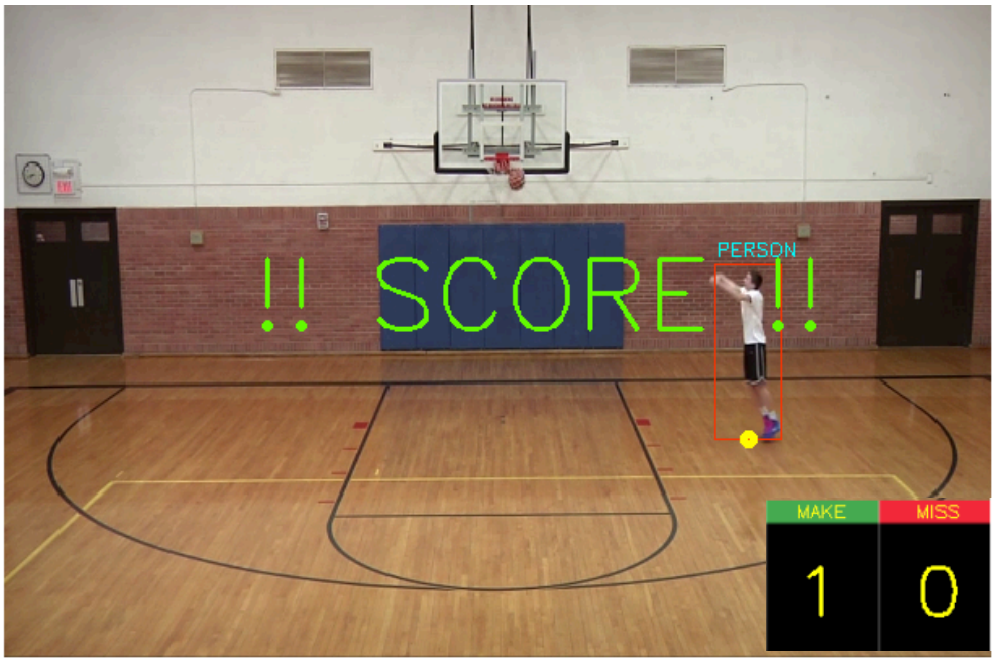
6.3.3 Player and Ball Detection Combined Functionality



6.3.4 Player position mapping

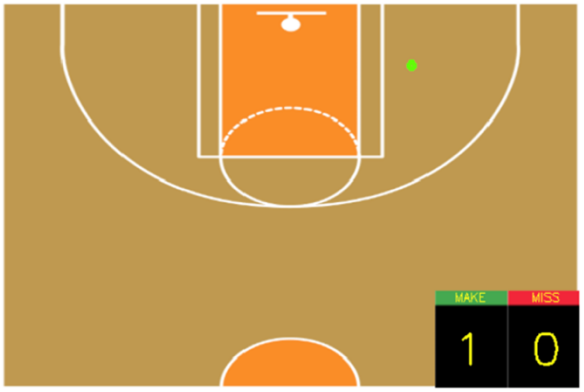


6.3.5 SSIM Test



6.3.6 Pose Estimator API

6.3.7 Final Application



7 Bibliography

Ananth, S. (2019, April 27). *R-CNN for object detection.* Retrieved from Towards Data Science: https://towardsdatascience.com/r-cnn-for-object-detection-a-technical-summary-9e7bfa8a557c

Bonasio, A. (2018, December 13). *Report: Global AI Research.* Retrieved from TECH TRENDS: https://techtrends.tech/tech-trends/report-global-ai-research/

En, J. L. (2019, Feb 24). *Intuitive Deep Learning Part 2: CNNs for Computer Vision.* Retrieved from towards data science: https://towardsdatascience.com/intuitive-deep-learning-part-2-cnns-for-computer-vision-472bbb2c8060

En, J. L. (2019, Feb 24). *What are Convolutional Neural Networks? How can we apply Neural Networks to recognize images?* Retrieved from towards data science: https://towardsdatascience.com/intuitive-deep-learning-part-2-cnns-for-computer-vision-472bbb2c8060

Gibson, A., & Patterson, J. (2017). *Deep Learning.* O'Reilly Media, Inc.

Goodfellow, I., Bengio, Y., & Courville, A. (2015). *Deep Learning.* London.

Gupta, N. (2019, October 8). *Why is Python Used for Machine Learning?* Retrieved from Hackernoon: https://hackernoon.com/why-python-used-for-machine-learning-u13f922ug

Kathuria, A. (2018, April 23). *What's new in YOLO v3?* Retrieved from towards data science: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

Press, G. (2016, December 30). *A Very Short History Of Artificial Intelligence (AI).* Retrieved from Forbes: https://www.forbes.com/sites/gilpress/2016/12/30/a-very-short-history-of-artificial-intelligence-ai/#35821d5b6fba

Sharma, P. (2018, December 6). *A Practical Guide to Object Detection using the Popular YOLO Framework – Part III (with Python codes).* Retrieved from Analytics Vidha: https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/