CourtCoach — Individual Performance Analysis

Gabriel Saleh, CSE. Cameron Weiss, CSE. Nectarios Ouzounidis, CSE. Michael Mouradian, EE.

Abstract—Performance analysis in sports, especially in basketball, is an increasing demand. The need for individuals to have access to their shooting and form data in an intuitive and efficient manner is what Court Coach addresses. It will allow players to readily evaluate their performance in a practice setting with limited setup and a user-friendly mobile application. Court Coach is designed to give control to the players so they do not have to set up a complicated system to improve themselves.

I. INTRODUCTION

In competitive sports, especially basketball, advanced statistics and analytics are more crucial than ever to developing and monitoring a player’s growth. Professional teams and well funded college programs have the resources to provide this data to their athletes and coaches, but it is almost impossible for the average player to access them. With Court Coach, our device will track a player’s shot location and shooting percentages while they practice on their own, whether in the gym or in the driveway.

There are currently some products that attempt to address simple statistics tracking on the court. A company called ShotTracker has developed a comprehensive analytic system that can be used by teams to analyze their practices. The system requires permanent camera installs, and expensive custom hardware, making it impractical for an individual to purchase and use. Another system that is closer to our design is an app called Homecourt. The app does not require any hardware, but has yet to be released and is unconfirmed to be reliable. In the past, the only way for a player to achieve this kind of data while practicing alone would be to manually track their shots with pen and paper, which is impractical and tedious.

The requirements of our design can be summed up into categories: Accessibility and Accuracy. A crucial component to our product is its ease of use. We are basing our design on a simple set up and practical user interface. The second requirement, accuracy, pertains to the level of precision that we are striving for when collecting data in order to make the design feel reliable and accurate. The numerical requirements in Figure 1 demonstrate the values we feel will provide that.

II. DESIGN

A. Overview

Since our system is going to track a player in real time there are three main components that we have split our project into and each will be outlined in greater detail throughout the report. First, we have our Shot Detection system which itself constitutes three components: (1) Ball Tracking, (2) Frame Mapping using Machine Learning, (3) Player Location Tracking. The second component is our Hoop Hardware which is used to keep track if a ball actually went in the hoop or not; this component is broken down into (1) Sensor hardware and (2) Data Transmission. Lastly we have our User Interface which is not broken down into subcomponents but encompasses all the data we have accumulated with our other components into a readable, user-friendly iOS application. Our block digram in Figure (x) gives the layout for how our sub-systems will interact.

The main piece of hardware and computing for our system is a Raspberry Pi 3 B+ (RPi) which itself has all the hardware specifications to meet our needs. With a port for a Camera that can interface seamlessly with the Raspberry Pi, an integrated Wifi module—2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth technology—4.2, BLE, and low power needs—5V/2.5A input, an ARMv8 64-bit SoC (1.4GHz) processor, and 1GB of SDRAM [1]. Each of these specs give us the flexibility for our design to work: the Raspberry PI camera makes it so we do not need to purchase a different proprietary camera and import that data from an external device, also it gives us the ability to set up the device courtside and capture the practice session in real time. The Wifi and Bluetooth technology will be used to connect to our iOS application and to the Hoop Hardware respectively. Its low power needs mean we can design or purchase a battery with a 5V/2.5A output that will make our design more comprehensive and portable. The Raspberry PI’s processor and memory give us the ability to run OpenCV and TensorFlow as Python packages which is integral to our image processing software. Our choice to use the Raspberry Pi makes the entire system more compact-able and cost-effective.

---

**Figure 1**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Detect Shot Taken</td>
<td>90% accuracy</td>
</tr>
<tr>
<td></td>
<td>Detect Made Basket</td>
<td>90% accuracy</td>
</tr>
<tr>
<td></td>
<td>Track Player location</td>
<td>Within 1 ft</td>
</tr>
</tbody>
</table>

G. M. Saleh from Framingham, Ma (e-mail: gsaleh@umass.edu).
C. P. Weiss from New Milford, CT (e-mail: cameronwiess@umass.edu).
N. P. Ouzounidis from Foxboro, Ma (e-mail: nouzounidis@umass.edu).
M. Mouradian from Wakefield, Ma (e-mail: mmouradian@umass.edu).
effective since the RPi itself is only $35. The assumption
being that any given user would not want to go through the
whole setup process on their own computer or laptop to get the
data and format it for an application it also would make the
app seem redundant if everything was already on a laptop.
Lastly we wanted a minicomputer that was easy to program
and allowed us to integrate our Python code with existing
hardware — the RPi’s operating system (Raspbian) gives us
that flexibility. Raspbian is a Unix based OS that lets us
import, edit, and save code either with an embedded GUI or
with SSH/SCP.

Using the RPi as our platform our main software
components will be constructed in terms of the components in
our block diagram. Shot Detection System means creating an
instance when a player takes a shot at any given location and
saving the data which includes the player’s location, snapshot
of their form, a timestamp, and whether the shot was a ‘miss’
or a ‘make’. This system relies a on couple of other sub-
systems and pieces of software that include the Ball-Tracking
software and the Player Location Estimation software along
with our Frame-Mapping technique. Once the ball has been
detected in the first frame and tracked throughout subsequent
frames we pass that information to the Player Location
software which will use the balls position in the frame to
estimate how far away the player is from the camera; that
software also requires some manual calibration to correctly
correlate pixel values to accurate distance values. Our frame-
mapping actually happens as part of the system software’s
initialization to give us a Y-value pixel to set as our threshold
to detect when a shot has taken place. Once our tracked ball
has risen passed our pre-determined Y-value we declare a shot
has been taken. Lastly, the player’s form has to be saved as
part of the bundle of data that will ultimately displayed on our
UI. All of this software is based on open-source OpenCV
Python libraries which is a big motivation for why we are
using Python as our primary language for software production.
OpenCV gives us a class of image processing functionalities
that are key to detecting and tracking the ball —HSV
colorspace detection. For our machine learning integration we
are using TensorFlow libraries to do much of the math
involved with Deep-Learning and Neural Networks. All of
these software components coalesce into our analytics for any
given shot a player makes which will be sent over Wifi to a
web server implementing an SQL database that has all of our
data regarding a player’s session. In terms of interacting our
Hoop Hardware the RPi will initially send a start message to
the hoop via Bluetooth at the start of a practice session, at the
end of a session the RPi will request the data from the Hoop
Hardware and congregate it with the rest of the data to send to
the database.

An important area of our software to discuss is our Player
Location Estimation. This part of the software determines
where a player is on the court so it then can map the shots they
take. Initially we had the idea to have the player wear a
Bluetooth tag in a wristband communicating with four
Bluetooth beacons spread around the court. The beacons we
chose were from a company called Kontakt.io which has uses
the beacons and a network gateway to track the locations of a
user in a building or outdoor area as a monitoring system. We
thought that using triangulation we could determine the
distance the player was from each beacon and in turn
determine where they are on the court. We decided against that
technology for two main reasons: (1) The setup time for the
beacon software was far greater than we anticipated and
required expertise we did not have and (2) we were not even
completely convinced the location estimation data coming
from the software gateway (which would have been one of our
phones) would have been accurate enough to meet one of our
aims which is to get the player’s shot location within 1’
accuracy. Those two things were enough for us to move
completely away from the beacon technology and use all
software to determine a player’s location.

Another main component that worked on and need to refine
is our Hoop Hardware. As of right now this consists of a
Raspberry Pi 3 B+, a LV-MaxSonar 1030 Ultrasonic Sensor,
an Analog-to-Digital converter, and a battery pack for power.

![Diagram](image.png)  
Figure 2
This setup has let us test the functionality of the Ultrasonic sensor to see if the beam it emits would be enough to capture the full width of the rim. So far our single sensor can capture most of the hoop with the outside edges being the exception. We plan to mitigate that problem with the addition of another sensor but place it in a way that it would not interfere with the current sensor. Our ultimate plan for the Hoop Hardware is to make it as compact as possible and replace the RPi with a smaller micro controller that fits our needs. As far as a working MDR deliverable our Hoop Hardware works as we need and is accurate enough that not much needs to change.

B. Shot Detection

I. Introduction

There are three things that go into shot detection which include: ball tracking, player location tracking, and machine learning for hoop detection. Ball tracking is the actual detection of if there exists a ball in the current frame, along with identifying the size of the ball and location relative to the x,y axis of the frame. The player location tracking algorithm takes the ball size from ball tracking algorithm and estimates the distance of the player based off of the size of the ball in the frame. The machine learning portion of the system is trained to detect where the rim is to improve calibration and accuracy of the actual detection of when a shot has been taken along with what to reference for the x-axis of mapping the players location. All of these things together creates a subsystem in the software that can detect when a shot was taken along with where the player is on the court.

II. Ball Tracking

The way that the ball is tracked in every frame is based off of the ball’s color. This idea was first inspired by a tutorial online by Adrian Rosebrock using the HSV (hue, saturation, value) colorspace to track a ball. The first step into doing this is creating lower and upper bounds for hue, saturation, and lightness such that only the ball would be shown in the contours of the image. If everything other than the ball is ignored, figuring out where the ball is and the size of the ball becomes possible. Originally the ball tracking was based off the HSV values of a brown/orange ball. The biggest issue with the brown/orange ball was that there are a lot of objects that can be near or on a basketball court that fall into this upper and lower bounds of HSV. This makes it hard to pick out the ball from other things going on in the frame. After running into this issue, next was using a neon green basketball. By using a distinct color ball that did not resemble anything except for random noise in the video, it was much easier to find the ball in the frame. Shown in Figure 3 Necto and Gabe with ball the neon ball is clearly shown using the HSV filtering with everything else in the frame completely blacked out. Once this filter is completed, the largest group of white pixels is found, and enclosed in a circle. The x,y coordinates of the center of the ball in the frame, in terms of pixels, along with the radius of the ball is now known.

III. Player Location

The player’s location is essential to analyzing shooting statistics. From the ball tracking methods, the size of the ball in the current frame along with the x-coordinate are given. Using this information, the players location needs to be mapped onto a basketball. The size of the ball is used to figure out the z-coordinate (depth from the camera to the ball) and the x-coordinate is used to figure out the x-coordinate directly on a basketball court, using the hoop as the x-axis origin. To figure out the z-coordinate using the size of the ball, there requires a step of calibration based on the known size of the basketball to find the size ratio to compare the size of the ball to. This calibration step is done using an image of the ball at a measured known distance of 10 feet away, with a known width of 28.5 inches, and, using the same HSV ball tracking algorithm before, calculate the size of the ball. The size of the ball, the known distance, and known width then creates a ratio that is used for all future size measurements to estimate the distance the ball is from the camera. This calculation is an estimate, and provides rough location, but with obstructed views of the player in the way and blurred images from a fast moving ball there is added error. The current implementation looks at the previous 15 distance calculations and chooses the minimum for the current shot being detected since most false estimations are of larger distances. This can be further improved using statistical analysis on the distance of the ball by removing outliers, analyzing direction of motion, and filters like the Kalman filter. The current implementation of the x-coordinate to player location methodology is much simpler. The pixels of the camera are linearly mapped to locations on the court. For the side of the camera with the hoop, wherever the hoop is, defines the origin of the court. The far opposite side corresponds to the half court line, and
the middle corresponds to right in between the middle. Given an x-coordinate of where the ball is in the frame, this is a direct mapping based on the defined space of the court. Currently this works well for locations in the middle of the frame but does not work on the far sides (hoop and half court) because camera’s do not create completely one dimensional images, instead there is an arc created by the lens. The current implementation can be improved by including this arc in the x-coordinate calculation using trigonometry instead of linearly mapping pixels to locations.

**III. Player Pose**

To aid in a basketball players development, shooting form can give direction for a player on where improvements can be made. Currently to do this, a burst of images is captured during the release of a players shot. To do this the program is constantly adding the most recent frame to a queue and removing the least recent frame keeping track of the most recent 15 video frames. Once a shot is detected, the least recent 5 frames are written to memory to allow for post-session analysis. This timing of which frames to save can be adjusted and will be tuned in the future, but for now gives a generally consistent burst of images at the time a shot is taken.

**IV. Frame Mapping with Machine Learning**

An indicator that a player has taken a shot in basketball is the height of the basketball. The maximum of a shot’s arc is nearly always well above the rim of the basketball hoop, so our team decided that we should set the trigger for the detected shot to be set at the height of the rim. That value will be determined automatically at the start of the session. The detection of the rim gives the ability to determine the X-origin for player location and the shot threshold value, with minimal amount of set up time.

To determine the location of the rim with a camera it is possible to use the HSV color space to determine that value, similar to what is mentioned in the previous section. After creating a black and white image, the average height of all white pixels that were found in the frame was determined. The issue with using the HSV algorithm to mask specific color ranges for the rim, is that it is not that distinct of a color. The algorithm was also finding pixels in other parts of the frame, which was affecting the determined height of the rim.

To mitigate these issues, our team decided to train a custom object detector to estimate the location of a backboard in the frame. The TensorFlow Object Detection API was used to accomplish this. We have started to gather images of basketball hoops in the area, and also images from the internet. These images were used as input to the training python script from the Object Detection API. The computing that these algorithms require are intensive. To run the trainer, we used a virtual machine with a GPU, through Google’s Cloud Computing. After training with different data sets, a couple different models were created. The models are used when running the object detection algorithm on pictures. Currently, these models vary on the pictures that they work for. One model has worked successfully for our testing inside a gym, detecting the backboard with at least 95% accuracy. The same model cannot accurate and consistently detect outdoor hoops. These issues could be resolved with more images to input to the training algorithm.

The ability to detect the backboard will help reduce where the rim detection algorithm has to look. The output of the object detection algorithm is a region of interest (ROI) with a confidence level of that there is a hoop there. Essentially our program currently crops the input photo to that ROI and uses the HSV mask as mentioned earlier. With this focused area the probability of a pixel belonging to the rim is higher than before, so the average is more accurate. Example (Figure 4) of our rim detection algorithm. The photo is cropped based on the backboard detection, and the line represents the height of the rim.

![Figure 4](image)

**C. Hoop Hardware**

Currently we have a Raspberry Pi B+ 3 connected to the back of the backboard that is powered by a portable battery. It is connected to the MB1030 LV_MaxSonar-EZ3 rangefinder. The rangefinder outputs an analog signal to a ADC that is then inputted to the Pi’s GPIO. The rangefinder is located under the rim where it connects to the backboard. The Pi is running a python script that is polling the sensor and sets a threshold that is equal to the distance to the back and front of rim. Therefore if there is a detection between those two distances it will count that as a score detected.

The plan for PDR is to get our hoop hardware to be 99% , and for us to put all the components into our PCB. Currently, we have tested score detection to be 90% accurate on makes. There were no false positives from missed shots, so we were 95% accurate overall. The missed detections were due to the ball being too far to the side. In order to mitigate this false negatives we will either add another sensor next to the current one, or add a similar sensor with a wider beam. The MB1030 LV-MaxSonar-EZ3 we currently are using has a higher side object rejection, as opposed to the MB1010 which is more sensitive around the edges of the beam. If this still does not remove the false positives we will resort to using two sensors side-by-side.

For CDR we plan on having the rangefinder, a microcontroller, and bluetooth module on a PCB. The
Team 5 Mid-Year Design Review Report, SDP19

The microcontroller will communicate with the rangefinder and process that data. The data will include a timestamp and whether a shot went in or not. The microcontroller needs to have enough memory to save that data for an entire session. The microcontroller will then transmit the data to the Pi using a bluetooth module. Eventually, the PCB should be enclosed in a case that can protect the hardware components from physical damage.

D. User Interface

The UI should provide users an effective way to access their statistics from their training session. Our plan is to develop an iOS app through XCode platform. Users will be able to check how they performed through the app after the session is over. The most significant part of the app are the two visual we would like to incorporate. Figure [5] shows an image of half of a basketball court. The shot location of a detected shot will be uploaded to the app to be displayed on a visual as is in Figure [5]. The X will represent a missed shot and a circle will represent a made shot.

III. PROJECT MANAGEMENT

This semester, we were able to satisfactorily achieve each of our three MDR goals: Shot detection, made basket detection, and player location mapping. These goals were set due to their importance as the three main parts of the system. We also accomplished a goal of ours that was not added to initial MDR goals: Pose detection. What is left for us to do is to combine these components into a complete system that sends and receives data between its different modules. There is also lots of refinement to do on each component of the system, as well as adding important features like the UI.

Gabe and Cam are doing much of the software with regards to shot detection and player mapping, with Gabe also building the hoop hardware prototype. Necto has been working on the frame mapping neural net, and has been contributing to the software and testing procedure. Mike has been assisting with testing and analytics, and is handling hoop hardware going forward. The team is doing very well in terms of communication and problem solving. Three of the team members are currently living together, allowing for quick answers to small questions and an easy place to get together and work on a major component.

IV. CONCLUSION

The group is very happy with our progress to this point and we have a solid plan for the coming months. Achieving all MDR deliverables is an encouraging step that we feel puts us on the right track to execute a great final design with consistent work in the winter and spring. Currently, with the...
project only existing in three distinct components, we are anticipating challenges when it comes to cohesive communications and reliability between components. The plan for the immediate future is to refine our three major systems to a point where we feel confident in their independent functions, which would allow us to focus on interconnection and more high level problems without worrying about individual components malfunctioning. Overall, the semester was a success in our eyes and we’re looking forward to future progress.

ACKNOWLEDGMENT

We would like to thank our advisor Professor Gao for the excellent support and advice in this first phase of the project.

REFERENCES