H.O.R.S.E Simulator

Final Report

Subrajit Surendran email: subr809@tamu.edu

James Robinson email: jakerdou@tamu.edu

Summary:

The H.O.R.S.E Simulator allows for users to play games on a basketball mini hoop and keep track of statistics like points, shots taken, etc. There are two modes that the user can choose from: Shootaround, and Multiplayer. Shootaround is a single-player mode in which the HORSE Simulator will keep track of shots attempted, shots made, and the distances that these shots were taken from. In the Multiplayer mode, two players can face each other in a game of HORSE. The HORSE Simulator keeps track of how far a shot needs to be taken from and which letter each player is on. Whichever player reaches the letter 'E' first is the loser.

The H.O.R.S.E Simulator is intended to be a useful tool for anyone who enjoys playing arcade-style basketball mini-games. There are three primary objectives of this project: 1) detect that a shot is made/missed, 2) detect the distance a player is shooting from, 3) log scores during the game. The main components of the project will be a raspberry pi camera module, a HC-SR04 ultrasonic distance sensor, and an RGB LED (anode). The camera will be used in determining how far away the player is from the basket. The ultrasonic sensor will be used to confirm whether a shot is made or not.

Introduction/Background:

Our proposed system offers a convenient tool for fans of mini hoop basketball and addresses a set of unmet needs. Most mini hoops don't have access to advanced shot tracking tools that are used today. Also, many shot tracking systems use some combination of machine learning and artificial intelligence and are only available at higher levels of play such as college and professional basketball. The system we are proposing provides players with a tool that has a high level of accuracy when it comes to keeping track of shots, the distance of these shots, and other statistics that will be beneficial to the player.

The second unmet need this project addresses is the lack of fun ways to play by yourself. The HORSE Simulator will allow players to keep track of their makes and misses and try to improve their scores. This provides a fun "game" to play that only takes one person.

Another unmet need is that players often don't know how to best improve the quality of their game. If a single player just stands around and shoots baskets, they will likely only shoot

shots that they are comfortable with, and they won't improve as much as if they had practiced shots outside their comfort zone. Our system will allow players practicing on their own to know which distance they are least comfortable shooting from, and to practice shooting from that distance so as to become a better player.

There are some environmental constraints to keep in mind during this project. This system will be designed for use in a flat, open area where the mini hoop can be placed. If the prototype is used outdoors, we need to ensure the camera is positioned in a way so that the sunlight does not interfere with its ability to detect the ball and player.

System Design:

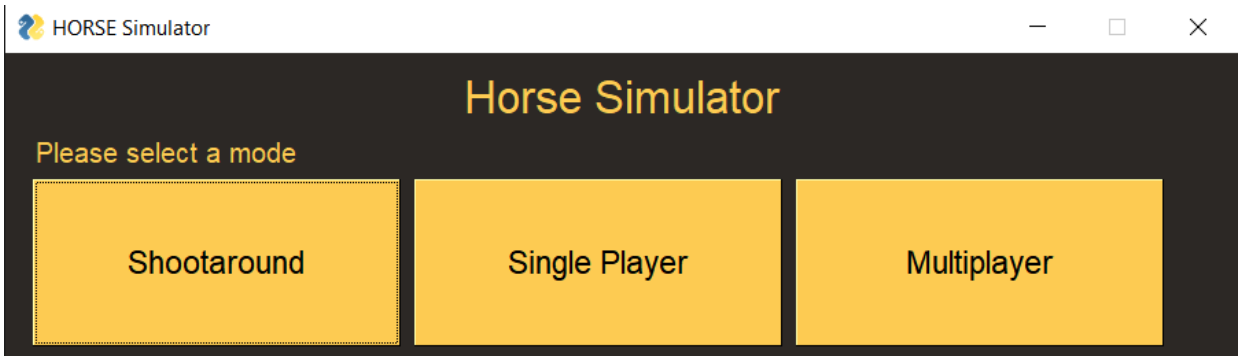
User Interface

The system will consist of a GUI that will be coded with Python. This interface will allow the user to select between the Shootaround or Multiplayer mode:

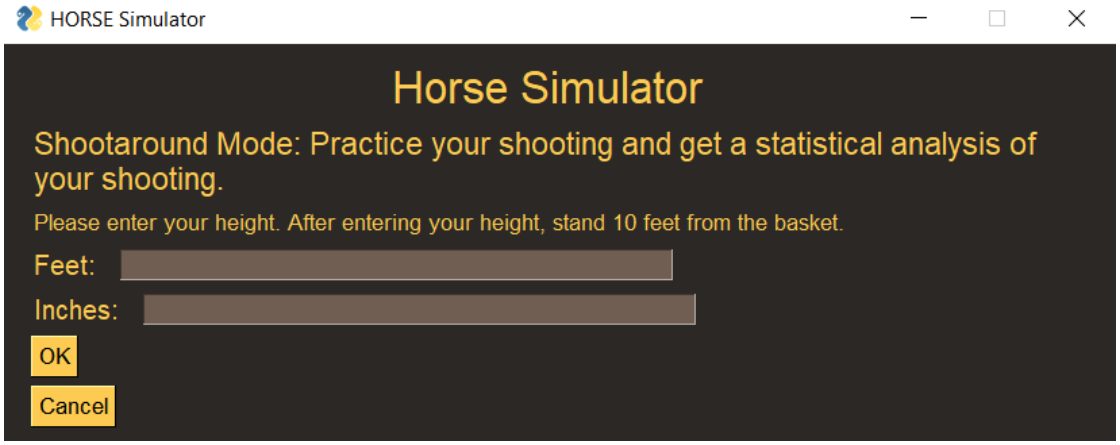
1) The Shootaround mode, designed for single-player use, will allow a player to work on their shot. This mode will be beneficial to the player who wants to focus strictly on shooting rather than playing a mini-game. The shot detection and distance detection features of our system will be used in keeping a log of shots the user has made/attempted and the distance each shot was attempted from. When the player is finished, they will select the "Finished" button on the interface and the interface will provide them with a view showing the total baskets made, total baskets missed, and other useful statistics that will help the player in improving their game.
2) The Multiplayer mode will simulate a real game of HORSE. The user interface will tell each player how far away to stand and will tell them when they are standing the correct distance from the camera and can shoot. The objective in the Multiplayer mode will be to beat the score of the other player. When the letter 'E' is reached by either player, the game will end.

Design Concept of GUI

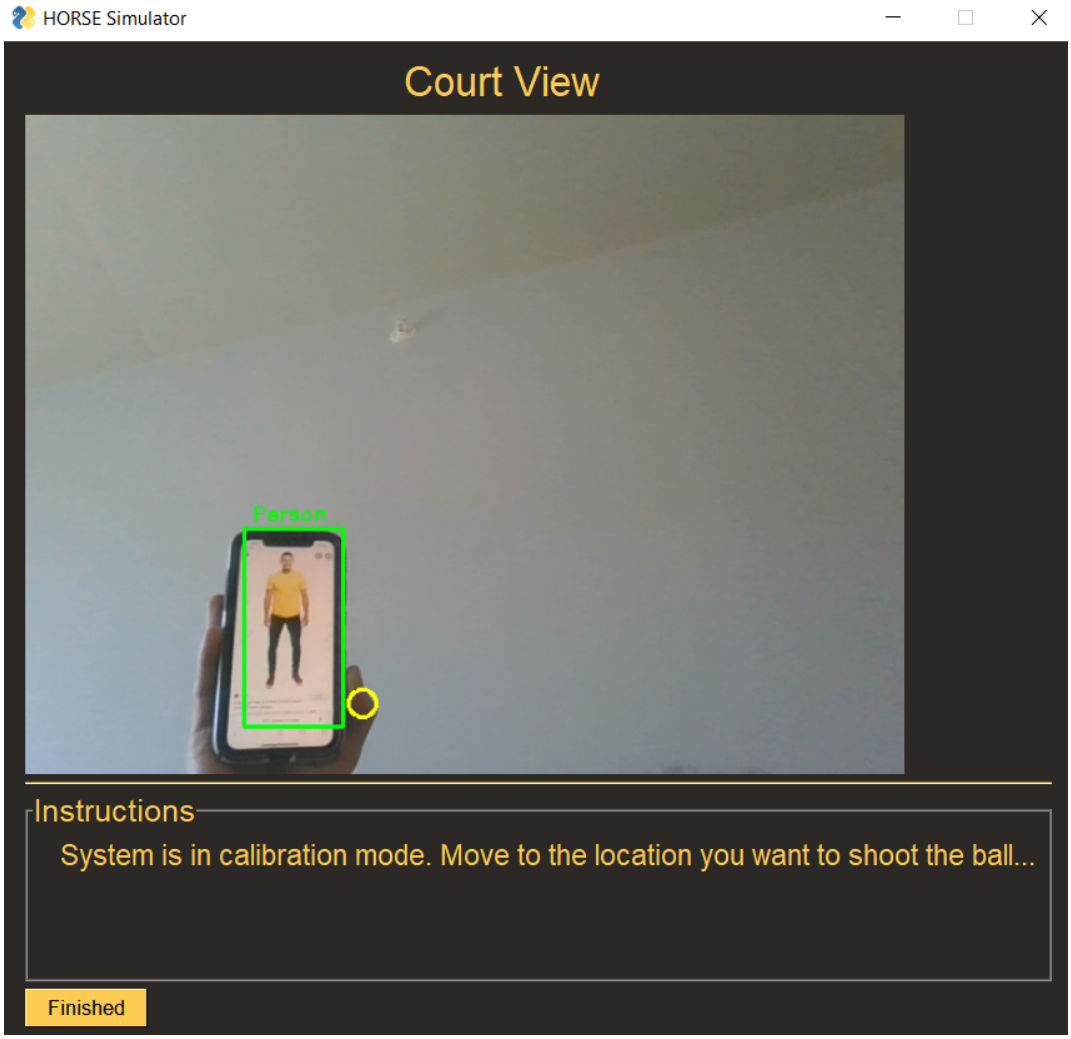
During development, we realized that the Single Player mode would be too similar to the Multiplayer mode and therefore redundant. We decided to not include it as a feature in the final product, however, you can still see the menu option in the screen captures of our GUI:



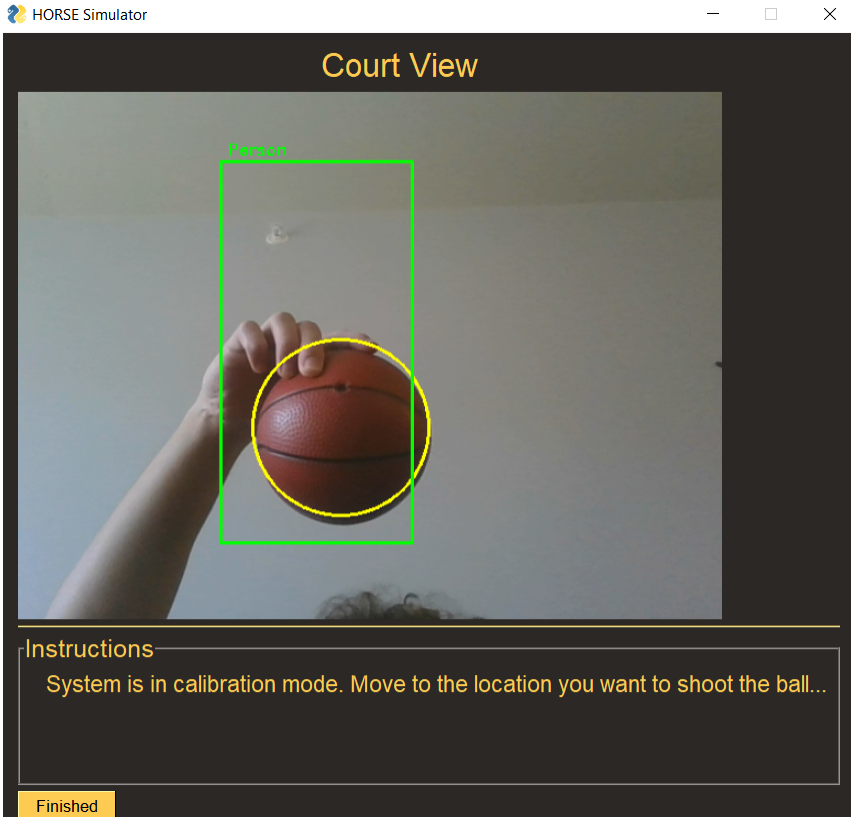
In the Shootaround mode, the user is first asked to enter their height and stand 10 feet from the system:



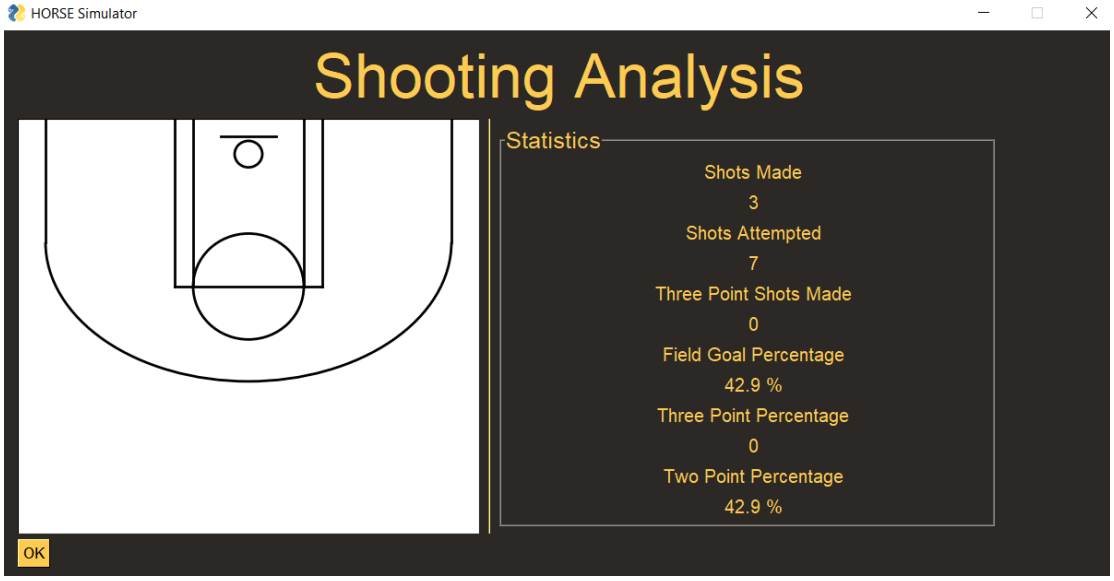
After the user has clicked ‘OK’, the following screen will appear and show the camera view as well as the shooting instructions. The system will then detect the person within the frame:



When the user is detected standing still for 5 seconds, the system will lock in on the frame of the person and begin to detect the ball. When the ball has been detected near eye level, the system will let the user know to shoot the ball and the user will have 5 seconds to shoot the ball:

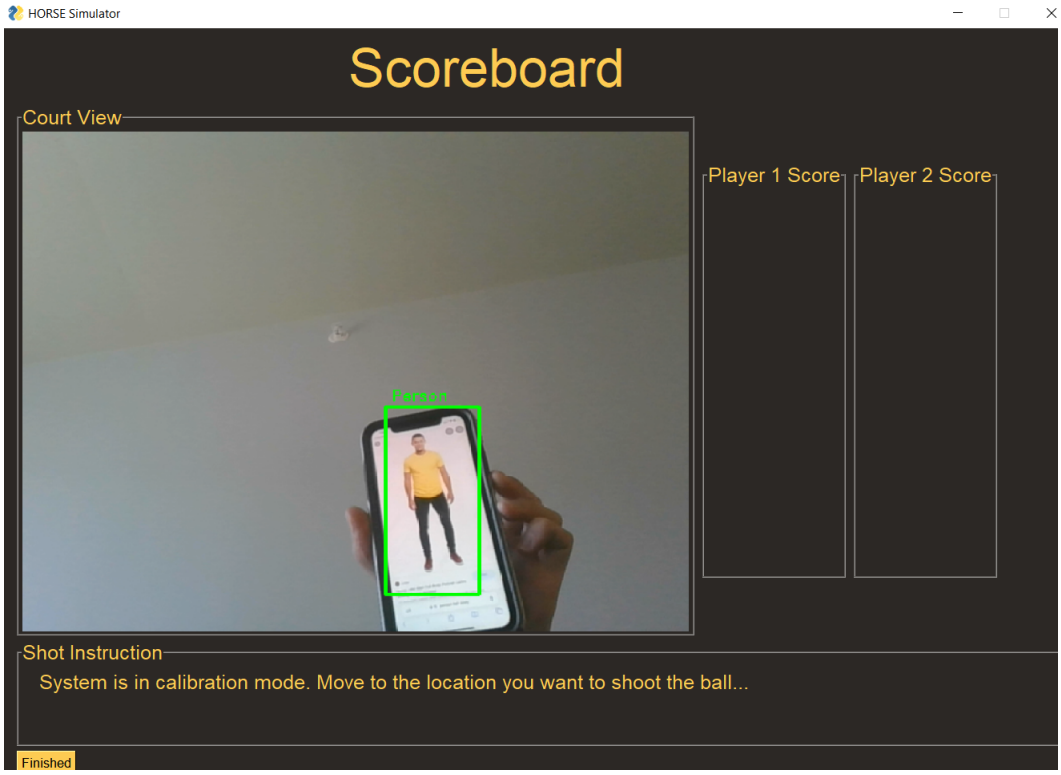


The final output will be a shooting chart showing how many shots the user has attempted and made:

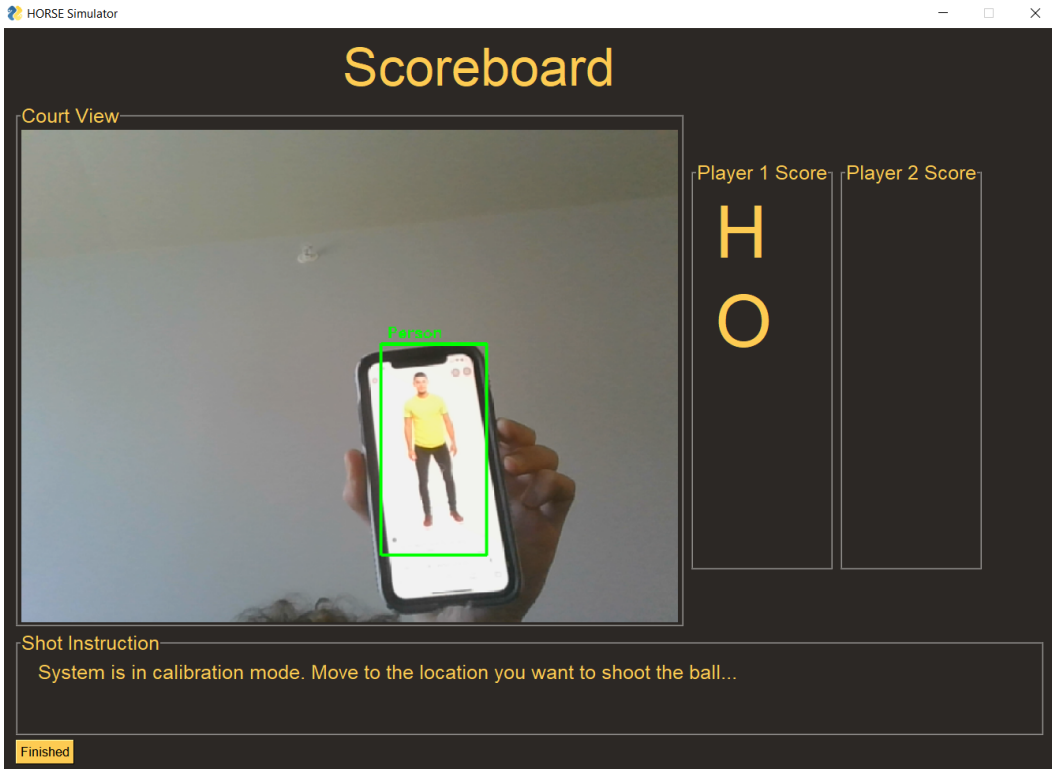


The multiplayer mode operates similarly to the shootaround mode. In the Multiplayer mode, each player is asked to enter their height:

When the users click ‘OK’, the next screen will show the camera view, scores of each player, and the shooting instructions:



Following the rules of H.O.R.S.E, the scores for each player will be updated accordingly:



Method of Person Detection

To detect people within the frame of the camera, our system uses OpenCV's HOGDescriptor Default People Detector. With this code already written, it was relatively simple to implement. The more difficult part was detecting when a person was standing still in the frame. To solve this problem, we kept a buffer of the previous ten frames, and within those frames, if a person with the same height was detected at the same location for five frames, we considered a person to be detected at that location and standing still.

Method of Ball Detection

To detect the ball, we used clever image processing techniques to filter the pixels in the image down to the ones that most likely included the ball. We did this by collecting the lowest and lowest HSV values of the ball, passing this information into our program, and removing all the pixels that did not fall within that range. The collection of pixels that remained was where we considered the ball to be detected.

Method of Shot Detection

To detect a shot, we waited until the person held the ball at about eye level. We did this by detecting a person, then waiting until the ball was detected within the top 12.5% of pixels that the person occupied. After the ball was detected as being ready to shoot, we listened to the sonar detector to decide whether or not the shot was made or missed. To do this we collected measurements from the sonar for seven seconds after the ball was detected at eye level. If within these measurements, 30 of them detected an object less than or equal to seven centimeters away (right in front of the basket), we considered the shot made. If not, we considered the shot as missed.

We had to close the bottom of the net of the HORSE Simulator. We found that if the ball passed through quickly like it would naturally, our sensor was not sensitive enough to produce reliable values and we would not be able to detect whether or not the shot was made.

Method of Distance Detection

To calculate the distance between the player and the basket, we asked the user to input their height (H), and to then stand ten feet (D) away from the camera. We then detected how many pixels "tall" they were (P). This allowed us to calculate the focal length (F) of the camera using the formula: $F = (P * D) / H$. Once we calculated the focal length, we were able to calculate the distance of the user no matter where they were in the camera's frame using the formula: $D = (F * H) / P$.

System Placement

The placement of the system was also a challenging aspect of this project. We placed the Raspberry Pi, breadboard with LED, camera, and sensor within a small, compact cardboard box and placed it directly behind the net to make the shot detection easier.

Design Concept for System Placement

Materials:

Component(s)	Cost/Source
Raspberry Pi 4	Cost: ~$40 (Already had) Source: Adafruit
Raspberry Pi Camera Module V1	Cost: $10

	Source: Arducam
HC-SR04 Ultrasonic Distance Sensor	Cost: $4 Source: Sparkfun
Basketball Mini-Hoop	N/A
3 1000 Ω resistors and 1 330 Ω resistor	N/A
Mini Breadboard, RGB LED, Wires	Source: Elegoo kit
Small Cardboard Box, Cardboard Pad, Tape	N/A

Appendix:

Resource List:

Ultrasonic Links

Sonar Distance

Ultrasonic Distance

Camera Links

Data Flair Human Detection

PyImageSearch Ball Tracking

Blog Electroica HSV Trackbar

Slack Contents:

November 5

Subrajit Surendran 3:20 PM

The only plan for this week was to order the materials for our project. We ordered the camera module for our Pi and have received it.

November 7

liu 7:54 AM

joined #horse-simulator.

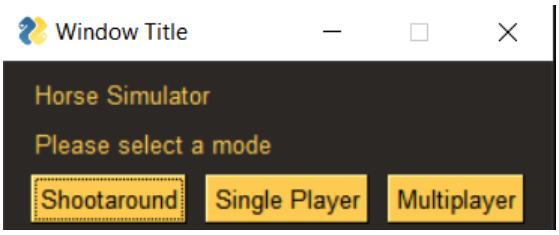
liu 7:58 AM

That is fine, but next time you should have some major progress.

November 11

James Robinson 5:42 PM

Our early stage GUI menu screen:



November 12

liu 8:39 AM

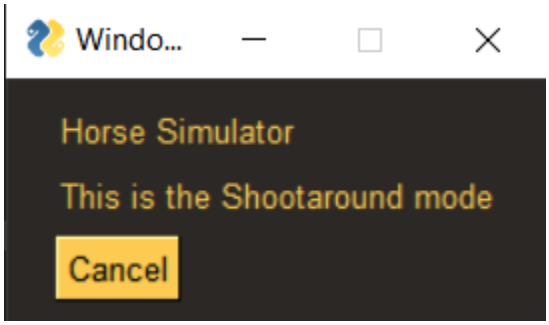
your project has very little activities. You will likely face a time crunch at end, when you realize your expectation may not be up to the standards.

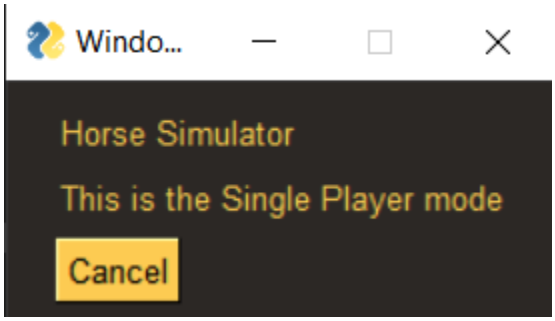
November 13

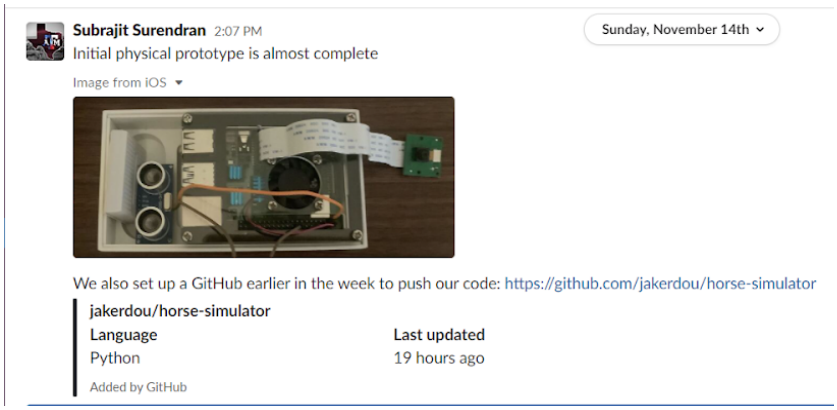
James Robinson 7:25 PM

added navigability to the GUI

2 files







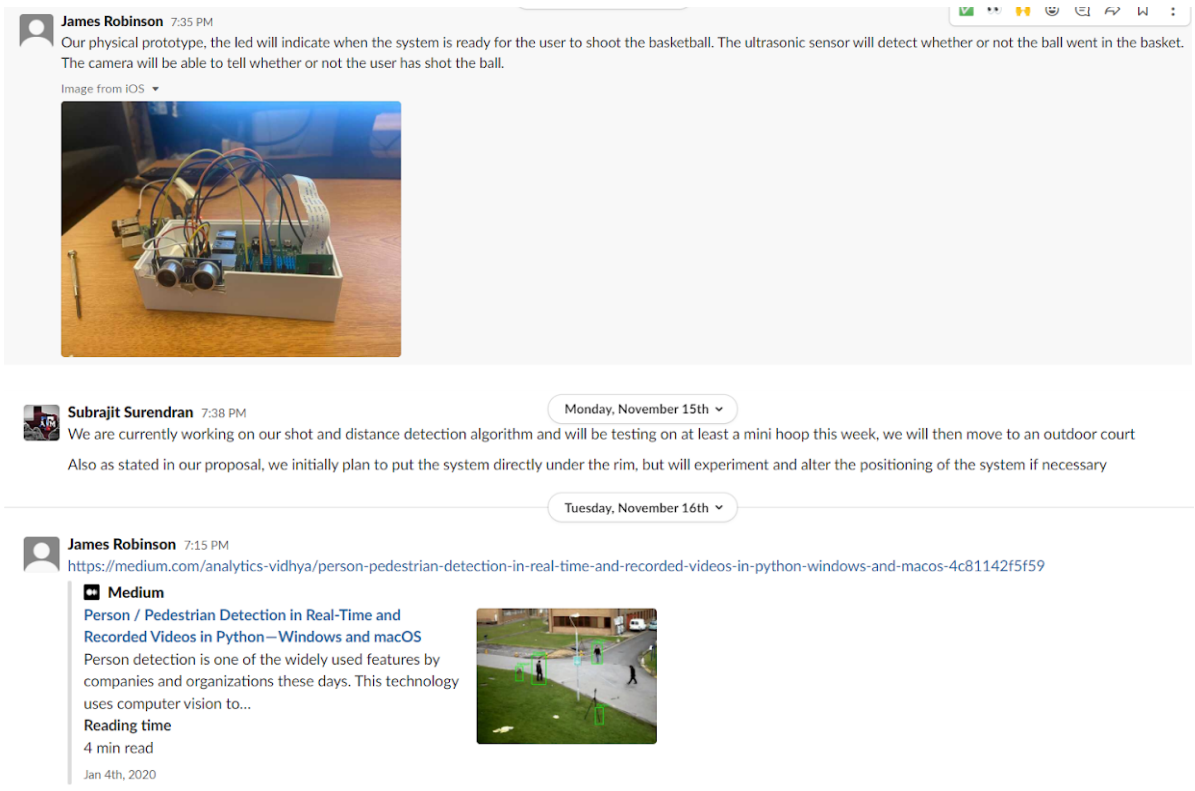
liu 2:48 PM

So you connect a few off the shelf modules to the microcontroller, and build a GUI so far. I still do not have a good sense of what your final system may look like

November 15

James Robinson 7:11 PM

https://learn.adafruit.com/ultrasonic-sonar-distance-sensors/python-circuitpython

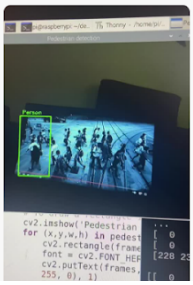


Jan 4th, 2020

Tuesday, November 16th

James Robinson 8:01 PM
Our program can now take input from a camera and detect people in the frame

Image from iOS



 # horse-simulator

5

James Robinson 8:22 PM
https://www.youtube.com/watch?v=FqK90JndaFk&t=292s

Tuesday, November 16th

YouTube | EndUser
Hong Kong - A Fast City in Slow Motion

Wednesday, November 17th

liu 9:24 AM
checked

James Robinson 12:38 PM
@liu, I am somewhat unclear as to what your expectations are for how we should report our work in this Slack channel. We are working very hard and I don't want to have any points taken off because we didn't document our work in a way that meets your standards.
I understand you are busy, but could you please inform us as to whether or not our progress reports in this Slack channel meet your expectations? If your expectations are not being met, please inform us in what ways we are lacking. Thanks

 # horse-simulator

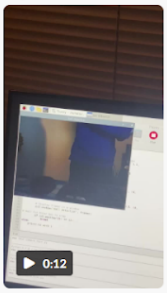
Thursday, November 18th

liu 8:54 AM
Your channel is not considered inadequate so far

James Robinson 10:34 AM
Thank you

James Robinson 7:24 PM
Our system can now detect balls

Image from iOS



horse-simulator

5

Jyh Liu 10:53 AM
joined #horse-simulator.

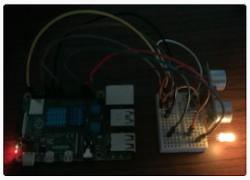
Friday, November 19th

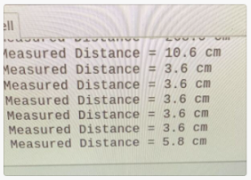
Jyh Liu 10:56 AM
glad to see your progress

Sunday, November 21st

Subrajit Surendran 11:14 AM
1) yellow light will be used in the beginning to indicate the system is in a calibration state and will switch to green when calibration is done
2) We will use the ultrasonic sensor to detect if the ball is within a certain distance from the system

2 files





GitHub has also been updated: https://github.com/jakerdou/horse-simulator

jakerdou/horse-simulator

Language	Last updated
Python	17 hours ago

horse-simulator

5

Monday, November 22nd

liu 3:18 PM
✅

Monday, November 29th

Jyh Liu 4:11 PM
very nice prototype

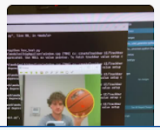
Tuesday, November 30th

James Robinson 11:55 AM
@liu @Jyh Liu Could you please tell us what the final due date for the project is? I cannot find it on the syllabus. Thank you.

Thursday, December 2nd

James Robinson 6:56 PM
Ball detection is much improved; more accurate and seemingly faster

Image from iOS



horse-simulator

5

liu 8:09 AM
So you are a single person team?

Saturday, December 4th

Subrajit Surendran 8:33 AM
No, we've split up the coding over the last few days. James has been working on the camera code, while I've been working on the ultrasonic and gui design. Today we will add the camera code to the gui file and be done with the gui implementation.

https://github.com/jakerdou/horse-simulator

jakerdou/horse-simulator

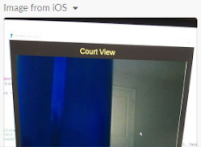
Language	Last updated
Python	15 hours ago

Added by GitHub

James Robinson 8:20 PM
The code can now tell how far away a person is

Subrajit Surendran 8:45 PM
Here are some of our GUI frames, each mode (shootaround, single-player, multi-player) initially asks the users for their height. Then there will be a transition to the court view in the shoot around mode and scoreboard + court view in the single/multiplayer mode. At the conclusion of the shootaround/single player game mode, there will be a shot chart that shows the misses and makes.

Image from iOS

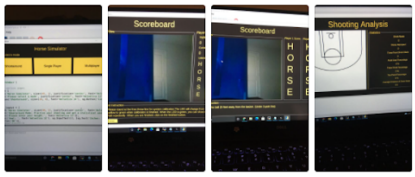


horse-simulator

5

Saturday, December 4th

4 files



The sonar has been tested on a mini hoop and works well, we will be testing the system on an actual court tomorrow (edited)

Sunday, December 5th

liu 2:38 PM
show me some live actions video

James Robinson 7:41 PM
We will have the video tomorrow

Monday, December 6th

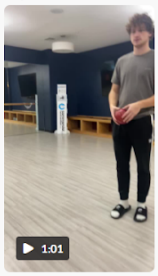
liu 7:49 AM
need to review today

horse-simulator

+ Add a bookmark

Monday, December 6th

James Robinson 1:23 PM
Image from iOS



liu 1:43 PM
good to see the demo

James Robinson 3:25 PM
@liu Subrajit and I both have something at 4, Subrajit won't be able to make the meeting at all, but I should only be occupied from 4-4:15. I will be in the zoom the whole time you are checking projects, but if you join our breakout room during this window, I may not be available