Autonomous Pong Robot
Project Final Report

Carlos Cuartas
University of Texas at Austin
Email: carlos.cuartas@utexas.edu

***Abstract*—Many robots nowadays haven't utilized the human ability to throw, launch, or toss. These actions are used by humans as a tool to extend reach for many possibilities and yet the majority of robots don't have the same capabilities. Robotics with machine learning paired alongside launching capabilities has the ability to reach a higher level of preciseness compared to humans. This can be proven with a simple trial of making a robot accurately launch a ball into a cup at many different distances. This is done with polynomial regression, a spinning wheel, and cup tracking. Keep reading to see how this robot function, what challenges were met, and its results.**

I. Introduction

In our day to day experiences, sometimes we need access to certain areas we can't reach physically. Like us, robots in all environments have a limit to how far it can reach as well. This is called their work envelope. Usually in order to increase the work envelope the robot has access to, the size of the robot increases as well. As the size of robots and their work envelopes increase, their price and dead zone increases as well. For humans, we compensate our short reach with throwing, tossing, or launching objects. Unfortunately, not many robots utilize this like humans do. This leaves us with a problem. If a robot can accurately increase its work envelope greater than its original working area, it could optimize factories, increase use of household robots, and open many new possibilities for robotics in general.

The overall plan has been to assemble a robot to shoot balls into cups using a spinning wheel and servo motors. It will be hosted on an Raspberry Pi 4 with an IRF3205 Mosfet H-Bridge, two PCA9685 boards, a DC motor, a basic web camera, an ultrasonic sensor, and some 3D printing to wrap it all together. The early stages of this project can be seen here in Figure 1.

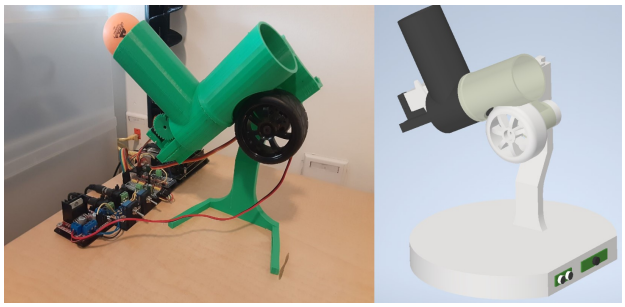

Fig. 1. Computer Assisted Design Launcher in early stages

This paper will explain the steps the project went through to go from idea to reality. It will cover the similar works, data gathered, methods, experiments, and the results of this project.

II. Goals

When this project was started, it was decided that the goal of this project would be to reach 70% success rate or higher. This was assumed due to the nature and uncertainty of lightweight projectiles. The robot must retain perfect accuracy through the vibrations of the 10,000 RPM motor, the inaccuracy of the 16bit PCA9685 servo controller, and the ball will even carry a spin that varies with how fast the motor spins when shot thus disrupting the accuracy of the shot. Even if all of the previous things mentioned function flawlessly the ball will still be affected by ambient airflow ever so slightly. More on the stochastic nature of this project will be discussed at a later point in this paper. So in total, the main goals for this robot will be as follows. It was also assumed that with longer and longer distances, the success rate would drop significantly. The distance this robot can shoot depends on mainly the sensors since the motor can shoot ping pong balls incredibly far. In order to keep it simple, it was decided that this robot needs to be able to work within a standard fold out table size of 200cm. A standard fold out table will have a width of 76cm so if the robot is at the end of the table it needs at least 20° of rotation to cover the table.

1) Make the ball in the cup 70% of the time a shot is taken and calculate the speed at which the motor spins with machine learning principles.
2) The cup can be anywhere between 0-200 cm of the robot.
3) The cup must be within a 30° angle of the robots facing direction.
4) The robot must be able to take aim at cups by rotating itself.

III. Literature Review

There is actually one paper similar to the concept of making balls into cups called "Robot Learning Project : Beer pong" by Johann Isaak[2]. Funny enough it describes how a robotic arm can acquire new skills, and specifically how to play beer pong autonomously. Other than that paper, there aren't many papers similar to the idea of this project but there are many related papers to individual concepts. Since polynomial regression will be a significant part to my project I will be referencing

some papers that are similar and some that have helped along with other papers with similar topics like tracking. Firstly there was a paper written by Eva Ostertagová called "Modelling using polynomial regression"[3]. This paper is concentrated on the application of polynomial regression models and has proven useful for determining when and how to apply certain equations to properly fit data. This paper is clearly similar to the section of polynomial regression but the similarities stop there. Another paper would be the "Robust Tracking in Low Light and Sudden Illumination Changes" written by Hatem Alismail, Brett Browning, and Simon Lucey in 2016 [1]. This helped with the robustness of tracking in all types of lighting so this project could track in many environments.

IV. Data

This robot will have two main functions for power calculation and tracking. First lets cover the power calculation data.

A. Power Calculation Data

In order to calculate the power to launch the balls we need data of previous successful launches. This data was gathered using a program made to do just that. This program will activate the ultrasonic sensor to determine the distance of the cup in front of the robot and use the power input into the program to test if the ball will successfully be launched into the cup. The program will then ask if the user to input if the launch was a success. If it was, the program will store the distance and power from the previous shot into a *.csv* list.

At times a shot will make it into the cup but due to the ultrasonic sensor being buggy at times, it would recorded the wrong distances occasionally. This could lead to problems when fitting the data so data scrubbing of vastly stray values was required. Here is an example of data before cleaning and after cleaning shown in the Figure below.

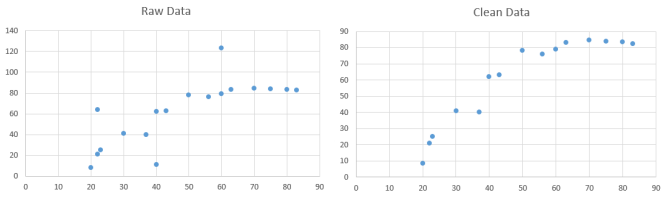

Fig. 2. PWM x Distance traveled for balls launched

The data gathered follows a polynomial regression model which is useful when there is reason to believe that relationship between two variables is curvilinear. The data between speed and distance in this case is curvilinear as shown in the Figure above.

B. Cup Tracking Data

In order to create a custom object tracking model one must first gather images of the object to track. For this project, only cups would need tracking thus requiring our data to be images of cups to track. A program was written to collect images from the web camera at set intervals and this was used in many different lighting conditions. A total of 100 images were collected and were labeled using a software called labelIMG. A custom model was trained using YOLOv5 from Ultralytics with the data for 200 epochs at a batch size of 16. The data used would look like the following Figure.

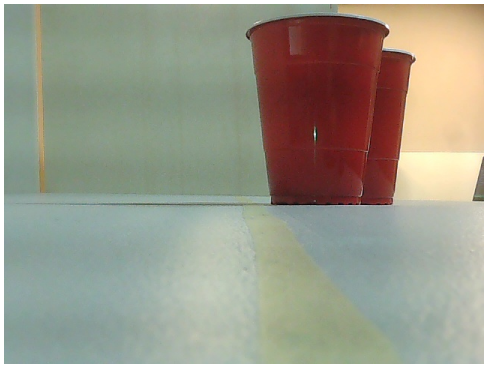

Fig. 3. Example Image Used for Training Object Tracking Model

Once all epochs were trained the model results were as follows in the Figure below.

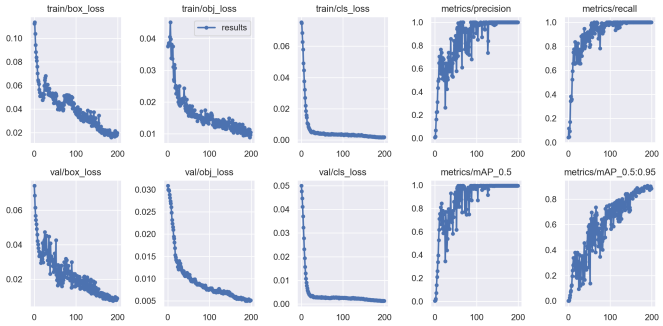

Fig. 4. YOLOv5 Training Results

V. Methods

In order to get the results needed, the robot first required a platform to run on with plenty of GPIO pins and fast speeds for object tracing. It is also needed that the platform is python based since the libraries required are PyTorch and OpenCV. It was decided that the NVIDIA Jetson Nano would be used due to its high performance and GPIO pin accessibility like the Raspberry Pi but with better CUDA support for computer vision.

Unfortunately, it was revealed that the Jetson Nano does not do real time processing when using the ultrasonic sensor. This causes all distances measured to become vastly unreliable. It was then decided to implement the Raspberry Pi 3b+ since it had a similar structure as the Jetson Nano and It was already owned. Unfortunately enough the Raspberry Pi 3b+ utilizes a 32 bit system architecture and PyTorch only functions on a 64 bit architecture. This then required the robot to be hosted on a Raspberry Pi 4 with a 64 bit operating system to be ran for object tracking. This board is shown in the Figure below.

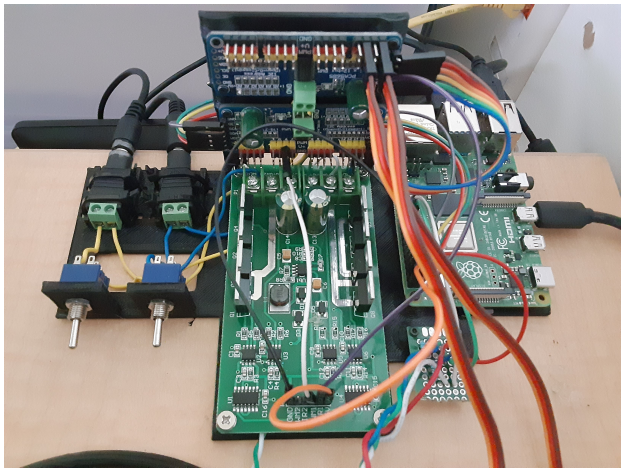

Fig. 5. Raspberry Pi Setup

In case it isn't known already, the Raspberry Pi is a low cost, credit-card sized computer that can be used just like the Nvidia Jetson Nano.

In order to launch the balls the robot is using a DC motor powered by an IRF3205 Mosfet H-Bridge in order to utilize the varying speeds PWM signals provide. To have access to a PWM signals, the pin for PWM from the board could be used but board can only control one PWM signal. In order to fix this problem I decided to use the PWM from the PCA9685 boards. Though this led us into another problem of frequency. The frequency dc motors run on are 400+ Hz at least but the PCA9685 still needs to power servo motors at a frequency of 50 Hz for the shooting and aiming. Unfortunately the board can only have one frequency at a time so I decided to chain two PCA9685s to hit both frequencies at the same time and power all servos and motors without restriction. This set up was demonstrated here in Figure 3.

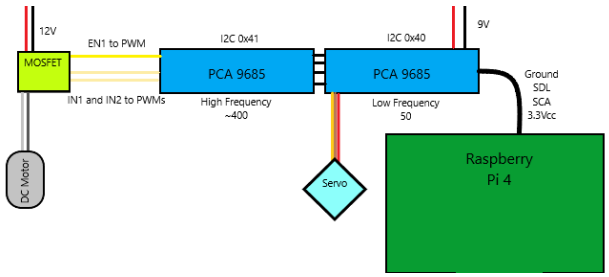

Fig. 6. Circuit Layout Plan for Electronics

The hardware was pieced together using many different types of boards which brings along a need of many different types of power sources. The robot required a total of 3 power sources to function.

- A 9V 2A power supply for the PCA9685 Boards and servo motors
- A 12-24V 6A power supply for the H-Bridge and DC motor
- And a 5v 3A power supply for the Raspberry Pi 4

This project will be divided into two main functions. The first will be the Power calculations and the other will be the yaw calculations.

1) Power Function: The robot will calculate the power needed by interpolating the power from the data gathered with the ultrasonic sensor data gathered earlier. By determining the distance from the ultrasonic sensor to the cup and comparing that distance to the few data points gathered beforehand it is possible to extract an accurate power value that corresponds to the distance of the cup. One way of doing this would be by using linear regression for this prediction. Though the problem with this is that the data gathered is not linear. Since this the data gathered doesn't fit with a linear regression expression but rather a polynomial expression, polynomial regression is needed as shown in the Figure.

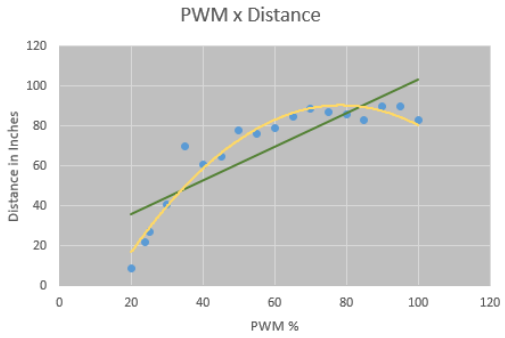

Fig. 7. Linear vs Polynomial Regression with data gathered

If the linear regression was used for this data it would be landing near the cup and almost never in the cups thus resulting in a lower accuracy rate. Thankfully polynomial regression can be applied on this data. The polynomial you see there has a degree of 2 but there could be a better fitting polynomial. In the Figure below, the degrees of the polynomial equations are increasing and yet its hard to determine which equation fits the best.

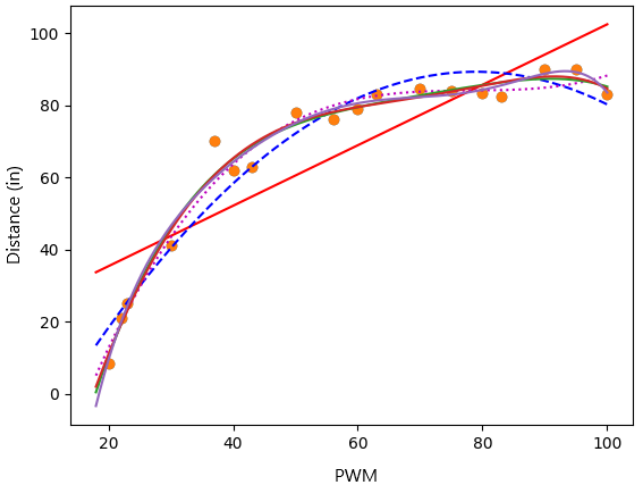

Fig. 8. Polynomial Regression with 1,2,3,4,5, and 6 Degrees

Its quite difficult to tell which degree best matches the gathered data but its clear that if the degree of the polynomial passes a certain threshold it will run into some severe over-fitting. So to determine the best fit possible, Bayesian Information Criterion (BIC) is applied. BIC will result in a value for every order used and to determine which degree best matches the data in question.

Bayesian Information Criterion

$$BIC_k = n * log(SS_\epsilon) + k * log(n)$$

k would be the number of degrees we want to test which in our case would be 6, n is the number of data points we have which would be 20, and SS_ϵ is the sum of squares of the residuals. The Residual sum of squares, or as many use it as RSS, is written out as follows.

Residual Sum of Squares (RSS)

$$RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

y_i is the i^{th} value of the variable to be predicted, $f(x_i)$ is the predicted value of y_i, and n is the amount of data. RSS is a measure of the discrepancy between the data and an estimation model. Typically, a small RSS indicates a tight fit of the model to the data thus its use being crucial to Bayesian Information Criterion. With that said lets look at the results BIC revealed.

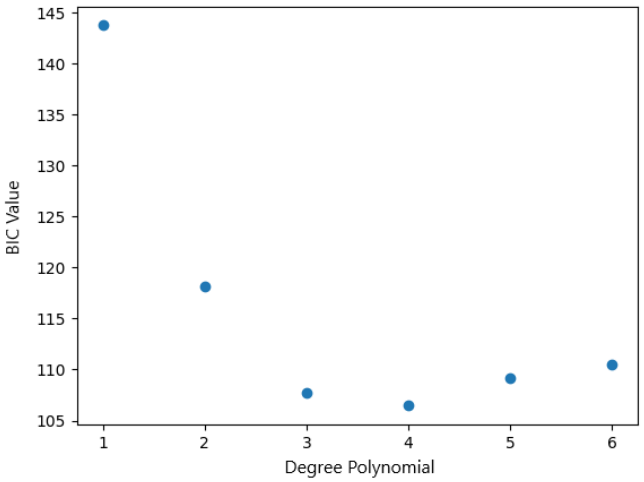

Fig. 9. Bayesian Information Criterion Results

When choosing from many degree models, usually models with the lowest BIC values are used. The Bayesian Information Criterion is an increasing function of the error variance σ^2 and also a rising function of k. Usually unexplained variation in the dependent variable increase the value of BIC but sometimes a lower BIC does not completely indicate one model is better than another unless a significant change in value exists. Because it involves a point system, BIC sometimes is a vague gauge to determine the best fitting model. It is also important to understand that BIC can be used to rank estimated models only when the data used is identical for all models being compared. Another thing about BIC is that it suffers from two main limitations.

1) Approximation is only valid for sample size n much larger than the number of parameters in the data being used.
2) BIC cannot handle complex collections of data in high-dimensions.

An example of a case where BIC couldn't be used in this project would be if the data gathered was so small that it matched with the amount of degrees the program was testing for. Thankfully this problem was averted with a simple use of more data points. With this said lets look at the Figure for BIC.The lowest value BIC is on the fourth degree polynomial. The third degree is a close second so either of these could be used, but for now the fourth will be chosen to run as shown in the Figure below.

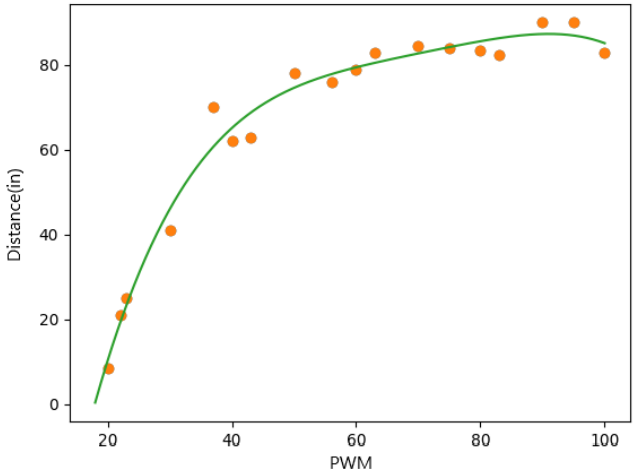

Fig. 10. Best Fitting Degree (Fourth Degree Polynomial)

Once the best degree to use is calculated from polynomial regression and Bayesian Information Criterion the program will use the interpolated polynomial line to calculate the power level from using the distance value from the ultrasonic sensor. This power value is sent from the Raspberry Pi to the PCA9685 to convert the power value to a PWM signal. The power is the level at which the PWM signal will operate at or simply, the Duty Cycle. The PCA9685 board will then send this duty cycle to the MOSFET H-Bridge to amplify the duty cycle to 24 volts to operate the DC motor at the desired power level. The motor is adhered to the wheel thus causing the wheel to rotate at the desired duty cycle. This is clearly demonstrated in the Figure below.

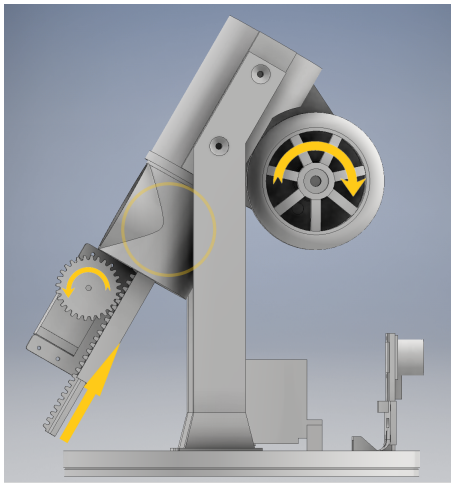

Fig. 11. Shooting Mechanisms

Once the desired speed is reached, the robot will send a signal to the servo motor through the PCA9685 to rotate the linear actuator forward powered with the servo motor. This will push the ping pong ball forward into the spinning wheel resulting in the ball being launched.

2) Yaw Section: This section is about the targeting in the yaw motion. The Yaw aiming function works by utilizing three things.

1) A Redragon GW800 1080P USB Webcam
2) One Servo Motor
3) And Object Tracking using OpenCV and PyTorch

The main purpose of this function is to point the barrel at the targeted cup by turning the base which is, in theory, a 3D printed lazy Susan with a geared servo motor attached to it to rotate it. The robot knows where to turn based on if the cup is to the right or left from the center of the robot when being tracked. The tracking initially consisted of a simple Hue Saturation Value color search using OpenCV. Once the object was detected the program would apply multiple closing kernels to remove false negatives and remove false positives. This looked like the Figure below.

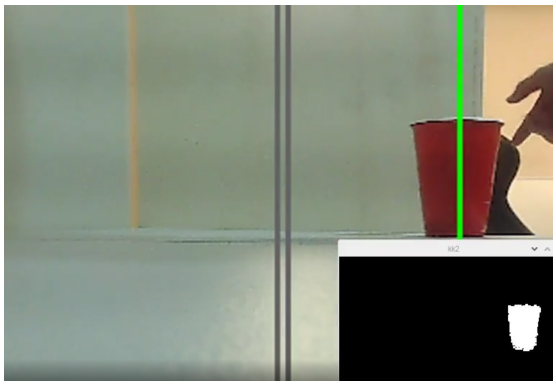

Fig. 12. HSV Tracking Using OpenCV

The green line will indicate the center of the cup and the gray two lines will be the threshold area of no action in order to keep the program from adjusting indefinitely. The robot will rotate clockwise if the green line is to the right and vice versa.

Unfortunately enough this method will not work in any other setting of lighting since it is not adaptive to color changes. This is due to the HSV method requiring a range of hue saturation and value to look for, but when the ambient lights change a small amount, the tracking fails. In order to fix this a custom object tracking model was made using PyTorch, OpenCV, and YOLOv5 from Ultralytics. As stated earlier, the data used to train this model was 100 images of cups in different lighting and different types of obstruction.

EXPERIMENTS

To determine how well this robot preforms, a series of experiments were tested. The first test was an accuracy test. This accuracy test was preformed with 25 launches at 7 different distances for the cups. To clarify, the steps taken to gather this data was to first position a cup at a set distance, then to run the autonomous launching program 25 times, and to mark each time a ball was made into a cup. This process was done for different distances seven times for a total of 125 launches. This data for the accuracy is shown below in the Figure.

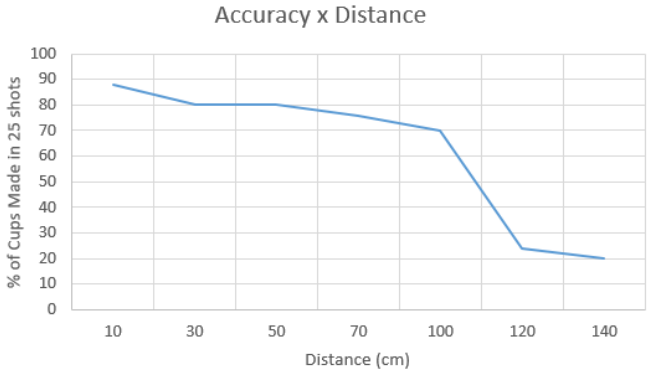

Fig. 13. Accuracy Results With Distance

As seen above, the accuracy is very high until it falls off significantly after 100 cm. It was understood from the beginning that this robot would carry some small errors into the launches. It was also theorized that with longer and longer distances introduced, the small errors would compound and slowly skew the preciseness of the launches at greater distances. Unfortunately, the relationship between the compounding errors and the distance was much higher than what was originally thought. This is shown clearly in the Figure below.

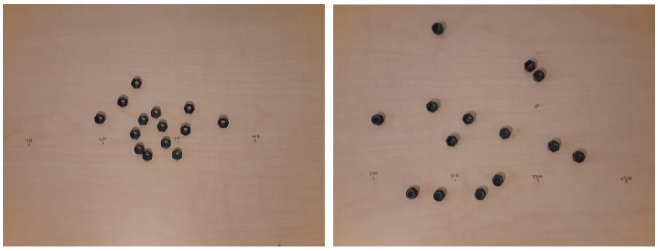

Fig. 14. Sparsity of Launches

As seen above the 30 cm distances has less of a spread compared to 110 cm launches. Standard deviation calculations were done in the distance axis and was determined that one standard deviation at 30 cm would be 4.4 cm compared to 7.2 cm for 110 cm. This caused long ranged shots past 100 cm to drop in chance of success, but within the 100 cm range nearly all shots would be made.

In order to run experiments on the cup tracking to see how well it preforms a multiple environments were created. These environments included bright, dim, and dark lighting along with obstructing and non obstructing environments. The results in terms of cup tracking were very surprising. These benchmark environments were only used to test the model and were not used to train on. The results are shown below for the benchmark environments while using the custom model made using Pytorch.

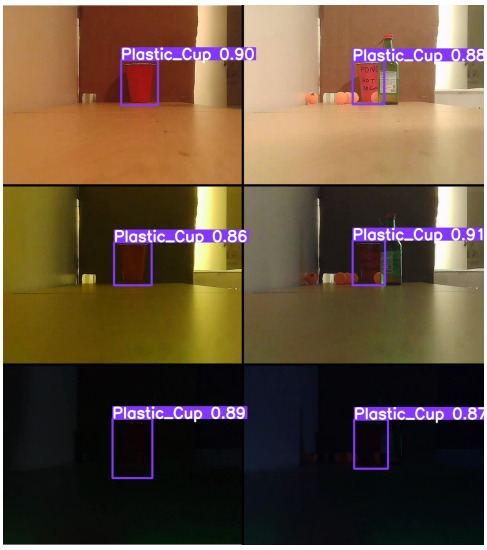

Fig. 15. Custom Model Trained in Multiple Settings

Surprisingly all cases passed with flying colors in every environment. The dark environments were used as a limit test and was not expected for it to track well at all but surprisingly it worked with no problem. Additionally the obstruction environment was challenging and expected to falter with some false positives the most but even in the darkest environment it passed.

Conclusion

The launching results were quite similar to what was originally hypothesized with a few exceptions. It was understood from the beginning that this robot wouldn't be perfect and that no matter what I did it would carry some small errors into the launches. It was also theorized that with longer and longer distances introduced, the small errors would compound and slowly skew the preciseness of the launches at greater distances. Unfortunately, the relationship between the compounding errors and the distance was much higher than what was originally thought. This caused long ranged shots past 100 cm very unlikely to make if first shot, although most people wouldn't be able to make a cup that far in five tries and much less in one. Nonetheless the results for the sub 100 cm distances were shocking as it worked with great precision and accuracy.

The cup tracking results were not what was expected at all. Humans struggle to see in the darkness and tracking in the darkness where little to no light passes would be nearly impossible. At least that was what was expected. This model trained surpassed all expectations in tracking with obstructions or no obstructions. As well in tracking in light, dim, and dark lighting. This was a success since even the combination of these hard to overcome problems were such simplicities for this custom cup tracking model.

Although there were many possible upgrades that could make this project better. Things like more refined launches, better preciseness, a Gaussian process on the data to predict power levels, better distance sensor, or even implementing reinforcement learning. I learned that if a small problem arises, not to overlook it but rather grind out all the possible solutions and to not be afraid to implement them even if it complicates things. Thanks to this project I learned a lot more about best fit models and how hard it is to gather, clean, and use data for machine learning.

In conclusion, after applying Polynomial Regression to a robot with perception and interaction capabilities, We can now successfully make any cup within a 100cm range on the first attempt and eventually make any cup within a 200 cm range after about 5 attempts and also it can now track cups in all lighting environments as well as even tracking cups with objects obstructing the way. I classify this robot as a success in proving that applying robotics with machine learning and launching should be a feature in more robots across the world.

References

[1] Hatem Alismail, Brett Browning, and Simon Lucey. Robust tracking in low light and sudden illumination changes. pages 389–398, 10 2016. doi: 10.1109/3DV.2016.48.

[2] Johann Isaak and M. Krönig. Robot learning project : Beer pong. 2013.

[3] Eva Ostertagova. Modelling using polynomial regression. *Procedia Engineering*, 48:500–506, 12 2012. doi: 10.1016/j.proeng.2012.09.545.

Fig. 16. Final Robot