T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

STRIKER

DERVİŞ ALİ DUMAN

SUPERVISOR
DR. ÖĞR. ÜYESI ALP ARSLAN BAYRAKÇI

GEBZE
2023

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

STRIKER

DERVİŞ ALİ DUMAN

SUPERVISOR
DR. ÖĞR. ÜYESI ALP ARSLAN BAYRAKÇI

2023
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 03/10/2023 by the following jury.

JURY

Member
(Supervisor) : Dr. Öğr. Üyesi Alp Arslan BAYRAKÇI

Member : Prof. Dr. Erkan ZERGEROĞLU

ABSTRACT

In these days, everything is slowly evolving into technology. As a result of its evolution into technology. Competition events such as Teknofest and so on are organized. In these and similar events, competitions are organized in line following robots, sumo robots, airplanes, drones and many other fields.

Slowly, sports fields are starting to be built with automation and one of these sport branch is football.

This machine detects the goal, processes the image and targets the goal. After aiming, the spring mechanism stretches and throws the ball into the goal. After the ball is thrown, we wait for him to understand whether there is a goal or not.

ACKNOWLEDGEMENT

I would like to thank Dr. Öğr. Üyesi Alp Arslan BAYRAKÇI who gave me the idea of this project.

DERVİŞ ALİ DUMAN

LIST OF SYMBOLS AND ABBREVIATIONS

HSV : **Hue, Saturation, Value Color ranges**
BGR : **Blue, Red, Green Colors**
SSH : **Secure Shell Host Connection**
VNC : **Virtual Network Computing**

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

In this project, I wanted to design a machine that detects the football goal and the ball from its color, and then combines the shoot parameters with the result of whether there is a goal or not with machine learning.
It is expected that machine will train itself with every shot it thrown by adjusting the angle, and gradually make more accurate shots with each shot through machine learning.

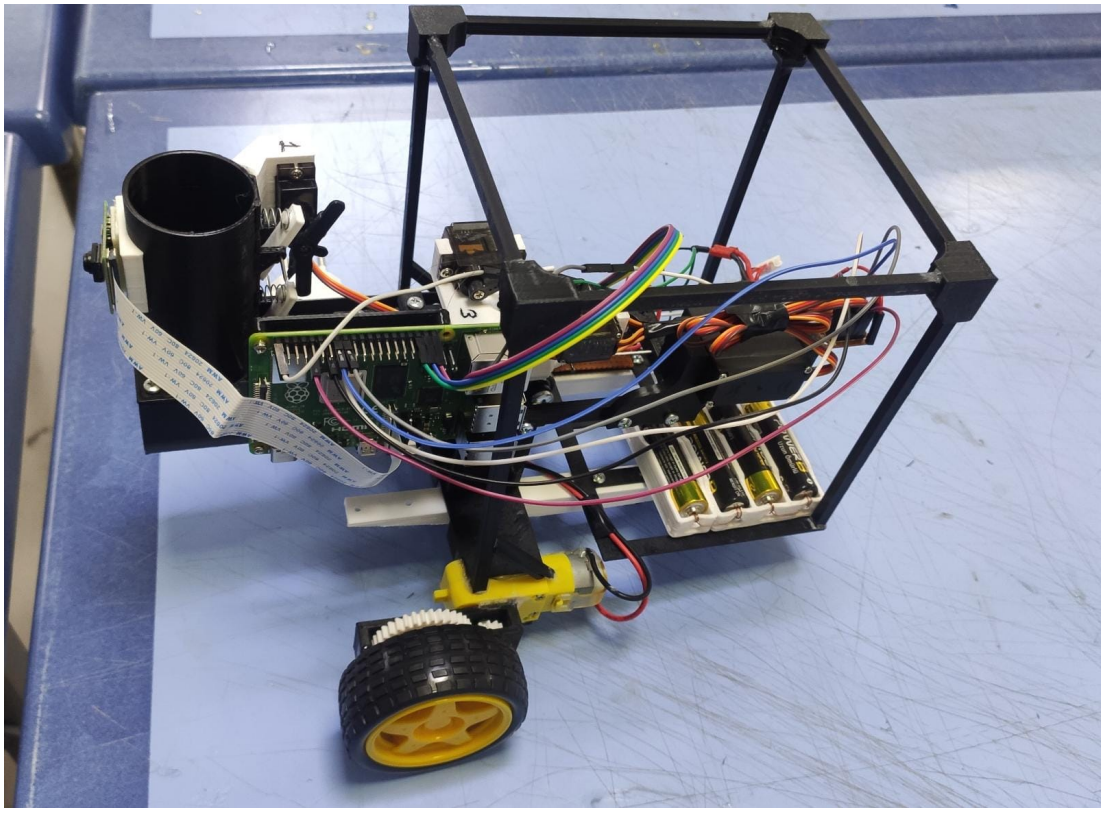

Figure 1.1: This is the complete machine of Striker project.

1.1. Working Environment

Project works on Raspberry pi 4 module B. All codes working on this environment. Make sure you are feeding the raspberry pi with the correct power supply (5V/3A). You must connect device through VNC server or you can connect the device through ssh connection. Project is in Desktop/Project file.

1.2. Used Technologies

1.2.1. Open CV

Open CV for image processing: recognizing of ball and soccer goal, if shot is successful or not, position of the machine and setting it the right angle. Measuring the distance of ball from soccer goal so that way we know which angle we must shot.

1.2.2. 3D Printing

Most of the mechanical parts used in the project were 3D printed. These 3D printed parts include battery and power bank slots, wheels for make the wheel turn slowly, a magazine that allows extra balls to be thrown, the servo in the last of the machine that allows the rope to be pulled at 90 degrees. In addition, football goals were taken with 3D printing.

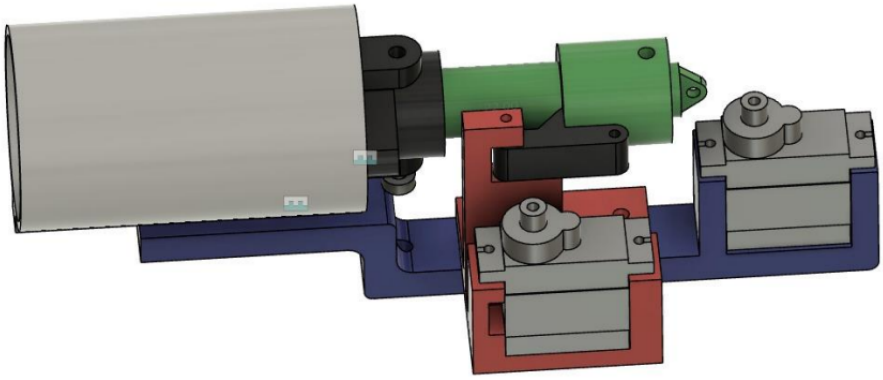

Figure 1.2: First version of my project.

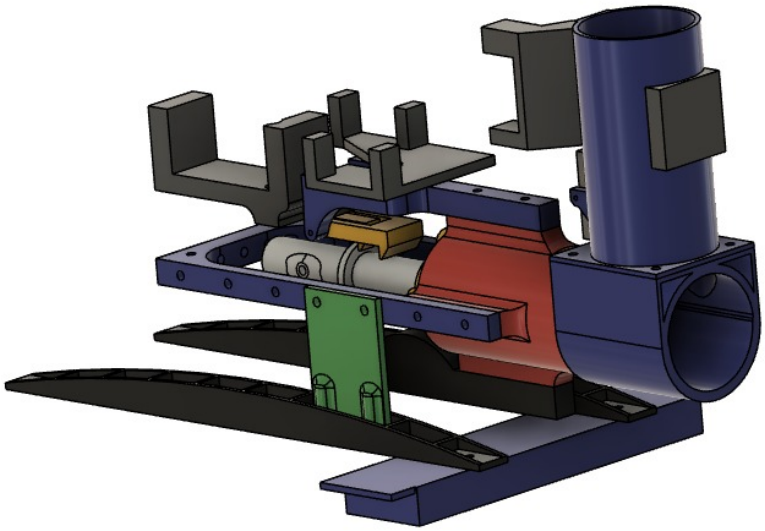

Figure 1.3: Current version of my project.

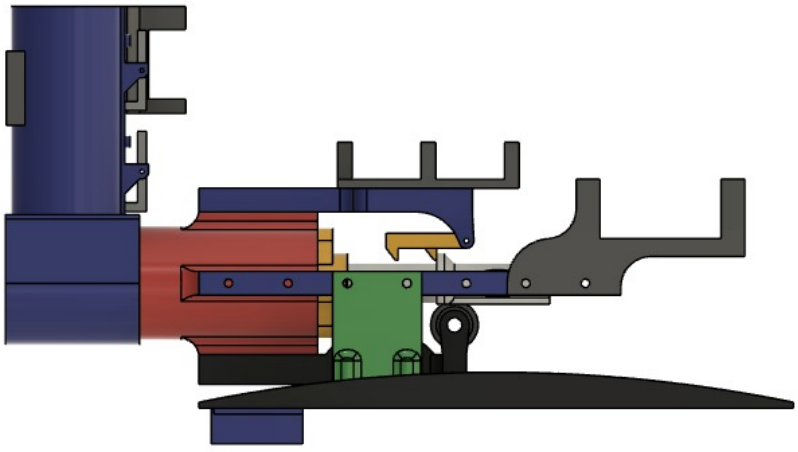

Figure 1.4: Current version of my project (Rotated).

2. IMAGE PROCESSING

2.1. Color Boundaries

Here I had 2 different options. The first was to search for objects in the color space with BGR, and the other was to search with HSV. My reason for choosing HSV is that my ball and pen can have different colors depending on the color, angle and intensity of the ambient light. In this case, it may be difficult to perceive the goal and the ball, or he may call different objects a ball or a soccer goal. To prevent this, I used HSV and gave a color range, not a certain color. The color space is created in a cylindrical shape with the color technology called HSV. In this way, we can choose the range we want. A Python script was used to find the HSV values. Thanks to this script, the color ranges received are included in the image processing code. I divide the H1 and H2 values in half and add the other values to our code as they are.

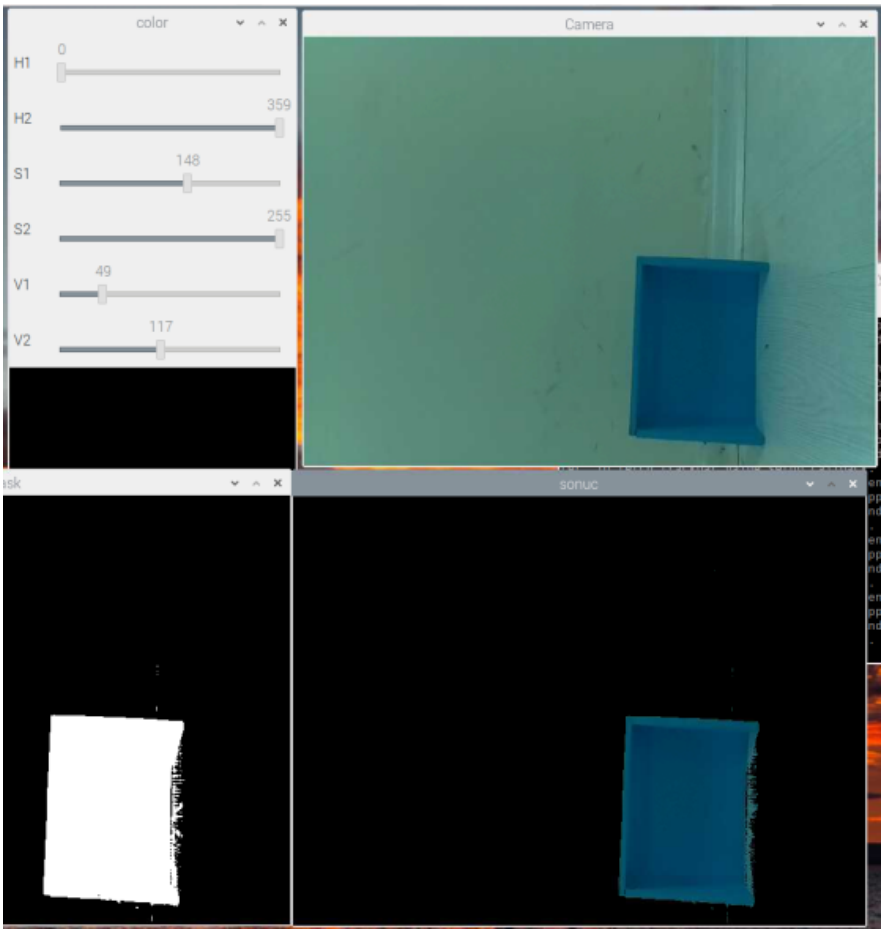

Figure 2.1: HSV Tracker.

2.2. Recognizing Soccer Goal

I blurred the received image to prevent it from detecting every colored object around. In this way, I prevent a little more from mistakenly selecting the wrong object for a soccer goal. Then, I define my color boundaries with the HSV values I found before and apply a color mask. From here, I find those areas using the find contours method to find the color-detected surfaces. I find my object through size of the object that looks like the largest rectangular shape, as a soccer goal, and then I draw a rectangle on the screen at the boundaries of the detected rectangle.

```
while ready == 0:
    ret, frame = cap.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

    # Find contours in the blue mask
    contours_blue, _ = cv2.findContours(mask_blue, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

    if len(contours_blue) > 0:
        c_blue = max(contours_blue, key=cv2.contourArea)
        x_blue,y_blue,w_blue,h_blue = cv2.boundingRect(c_blue)
        position_goal_x = int(x_blue + w_blue/2)
        position_goal_y = int(y_blue + h_blue/2)
        cv2.circle(frame, (position_goal_x, position_goal_y), 5, (0, 0, 255), -1)

        # Check if the rectangle is within the desiblue range
        if min_length < w_blue < max_length and min_length < h_blue < max_length:
            # Draw a rectangle around the largest contour
            cv2.rectangle(frame, (x_blue, y_blue), (x_blue + w_blue, y_blue + h_blue), (0, 0, 255), 2)
            position_x = frame.shape[1]/2 - x_blue - w_blue/2
            position_y = frame.shape[0]/2 - y_blue - h_blue/2

            if position_y < -5:
                MOTOR_turn_left()
                ready = 0
            elif position_y > 5:
                MOTOR_turn_right()
                ready = 0
            else:
                MOTOR_stop()
                ready = 1
    else:
        MOTOR_turn_left()
```

Figure 2.2: Soccer Goal recognizing algorithm.

Figure 2.3: Soccer Goal recognizing.

2.3. Recognizing Ball

I have done similar operations (blur, HSV boundaried color mask, finding contours) here as in finding the soccer goal. If the matching image is a red ball, it perceives this image as a ball. After detecting the ball, it remains to understand whether there is a goal or not.

2.4. How code understands if it is a goal or not?

There are propotions of the soccer goal so I have to calculate the soccer goal's propotions. Then I have to compare contours of the objects (goal and the ball) So I can decide if the ball is in the goal or not. This image must be recognized so fast. It works back to back with the servo.h library because while throwing the ball with servos, it is expected to take that image instantly and decide whether a goal is scored or not.

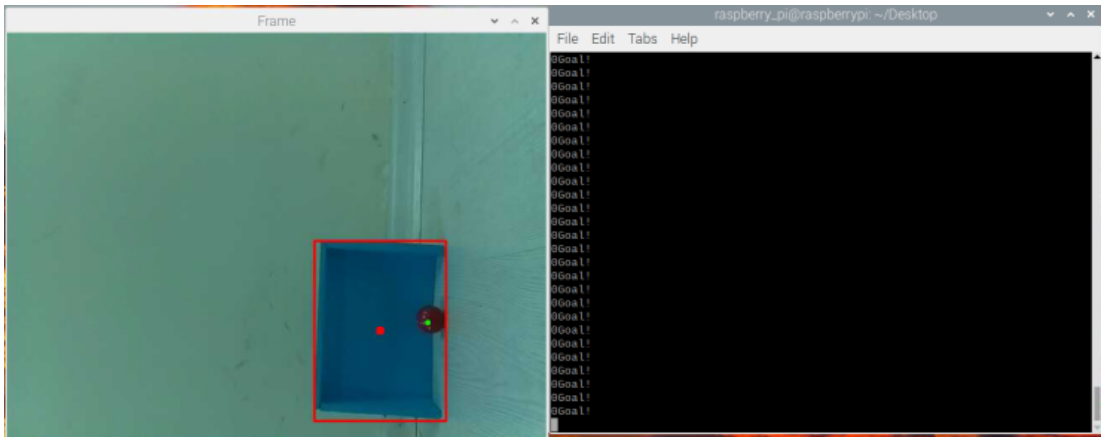

Figure 2.4: Goal or Not.

3. MACHINE LEARNING

I can not recognize the ball from 4 meters away so I cannot train the machine. Machine learning improves the shots but not improves that as we wanted. First values are first estimated values and the second values are the results.

3.1. Used Algorithm

Since I use raspberry pi, I have to use most optimal algorithm. So In my project, I used linear regression to develop a model for making more accurate shots in soccer. The main goal of the project was to train the model to predict the optimal angle at which a shot should be taken in order to score a goal by hitting the center of the goal.

I trained the model on a set of data that I collected and it was able to accurately predict the optimal angle. In order to improve the accuracy of the model.

The results of this project show that linear regression can be an effective tool for improving the accuracy of soccer shots but it is not effective as we expected, and that by considering various factors that may affect the outcome of a shot, it is possible to achieve a high degree of accuracy in predicting the optimal angle at which a shot should be taken in order to score a goal.

3.2. Algorithm Design

The code calculates the coefficients (m and b) for a linear regression model. The algorithm works as follows:

1. The data is read from a source, which is assumed to contain two columns - one for the independent variable (x) and one for the dependent variable (y).

2. The number of data points (n) is calculated by finding the length of the independent variable data.

3. The slope (m) of the regression line is calculated using the following formula:

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

4. The y-intercept (b) of the regression line is calculated using the following formula:

$$b = \frac{\sum_{i=1}^{n} y_i - m \sum_{i=1}^{n} x_i}{n}$$

5. The function returns the calculated values of m and b as the coefficients of the linear regression model.

3.3. Algorithm Implementation

```python
def calculatePredict():
  data_y = [i[1] for i in read_data()]
  data_x = [i[0] for i in read_data()]
  n = len(data_x)
  m_top = n * sum([data_x[i]*data_y[i] for i in range(n)]) - sum(data_x) * sum(data_y)
  m_bottom = n * sum([data_x[i]**2 for i in range(n)]) - (sum(data_x))**2
  m = m_top / m_bottom
  b = (sum(data_y) - m * sum(data_x)) / n
  return m, b

def predict(goal_x):
  m, b = calculatePredict()
  return m * goal_x + b
```

Figure 3.1: This is linear regression algorithm used in my code.

This is a function written in Python to calculate the coefficients (m and b) for a linear regression model. The function first reads in data from a source (presumably a CSV file or some other data storage), which is assumed to contain two columns of data - one for the independent variable (x) and one for the dependent variable (y). It then calculates the coefficients. Finally, the function returns the coefficients m and b.

4. SUCCESS CRITERIAS

- **Shots can be made different size of goals.**
- **Shots can be made different angles and distances.**
- **Shots will have %80 accuracy.**

5. CONCLUSION RESULTS

In this project, I developed myself in 3 different directions. On the mechanical side, I learned how to design and 3D print a robot. On the electronics side, I learned how to set up a circuit, how to create a circuit board, and how to distribute power. On the software side, I learned how to process the image, what I can do with the image I processed, what are the machine learning and algorithms.
Machine can not realize the ball until 4 meters So it can not train the machine. Also machine algorithm cannot efficient as we expected. I couldn't change the algorithm because my raspberry pi is not powerful enough.

5.1. Results

It can shoot at different castle sizes and from different distances and angles.
Machine algorithm optimizes shots but it is not powerful as we expected.
There are 3 soccer goal size. The results for each goal size:

Big Goal:
It can get a goal from 3 meters %95 accuracy
5 meters %90 accuracy,
from 8 meters %50 accuracy

Medium Goal:
It can get a goal from 3 meters %90 accuracy,
from 5 meters %75 accuracy,
from 8 meters %30 accuracy,

Small Goal:
from It can get a goal from 3 meters %70 accuracy,
from 5 meters %40 accuracy

[1]–[3]

BIBLIOGRAPHY

[1] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A brief introduction to opencv," in *2012 proceedings of the 35th international convention MIPRO*, IEEE, 2012, pp. 1725–1730.

[2] G. Xie and W. Lu, "Image edge detection based on opencv," *International Journal of Electronics and Electrical Engineering*, vol. 1, no. 2, pp. 104–106, 2013.

[3] D. Maulud and A. M. Abdulazeez, "A review on linear regression comprehensive in machine learning," *Journal of Applied Science and Technology Trends*, vol. 1, no. 4, pp. 140–147, 2020.