BUILDING A RANGE FINDER SENSOR

DECEMBER 5, 2021

Submitted to: Prof. Dr. Kemalettin Erbatur

Name of Student: Moses Chuka Ebere
Student Number: 31491
Course: Autonomous Mobile Robotics (ME 525) Lab

TABLE OF CONTENTS

INTRODUCTION

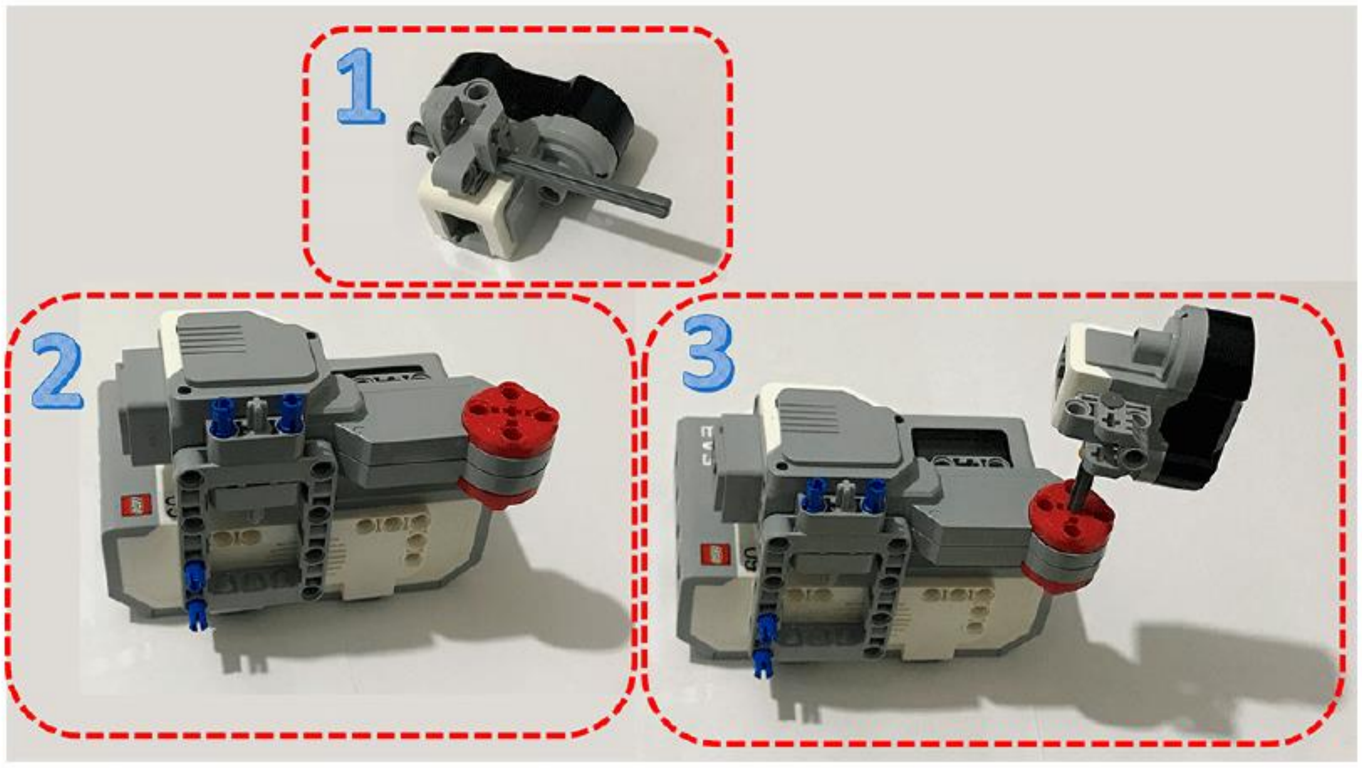
This lab involves a task to build a range finder using the EV3 toolbox.

The major aim of this lab is to explore the performance of an active ranging device in detecting distances from obstacles. Basically, we apply the principle of reflection of infrared light from solid objects.

Rotary motion of the sensor is achieved using a motor, and with the aid of a predetermined coefficient, sensor readings are converted to metric information, and resulting plots are produced. The experiment helps us identify the IR sensor as an exteroceptive, active sensor.

PROCEDURE

Firstly, the range finder is assembled using the necessary parts as shown in the figure below:



The major components used are:
- One large motor - One infrared sensor
- The EV3 Brick

To ensure that the infrared sensor rotates smoothly with the motor, a sturdy cord is used to attach it to the large motor.
MATLAB is the underpinning software for this application.

To implement rotary motion of the infrared sensor using the motor and plot the resulting x and y coordinates, a program is written in MATLAB. Prior to this, a USB connection is established between the robot and the PC on which the MATLAB code is written.

Pre-Coding Step

In the proximity mode, the infrared sensor reports the distance using values between 0 (very close) and 100 (far away), not as specific metric measurements. Therefore, a coefficient is required to convert infrared measurements to centimeters.
For this, an A4-sized paper (length 297mm) is placed between the sensor and a large black object. An extra 3mm is kept between the A4 and the sensor. The proximity is read by the sensor using the “readProximity” function.
The result obtained was 66. Therefore, **the conversion coefficient is 30cm/66**.

MATLAB Code

After creating a connection to the Ev3 and determining the conversion coefficient, the motor and IR sensors are properly defined.

- Arrays (x, y, d and θ) are created to store the necessary data for the orientation, distance, x- and y- coordinates.
- A counter (i) is created to facilitate storage of data in the aforementioned arrays.
- The motors encoder is reset, and the motor is run with a user-defined speed (e.g., at 20%).
- A while loop is used to rotate the motor for some time for proper data collection.
- Within the while loop, the direction of the IR sensor (with respect to the x-axis) is obtained as the encoder value. Using this θ, a set of if statements (2, to be precise) are included to ensure that the motor only rotates between 0 and 180 degrees.
- The distance reading is now retrieved from the IR sensor, and it is used to calculate the coordinates together with the θ.

  $$x = d\,cos(\theta), \quad y = d\,sin(\theta)$$

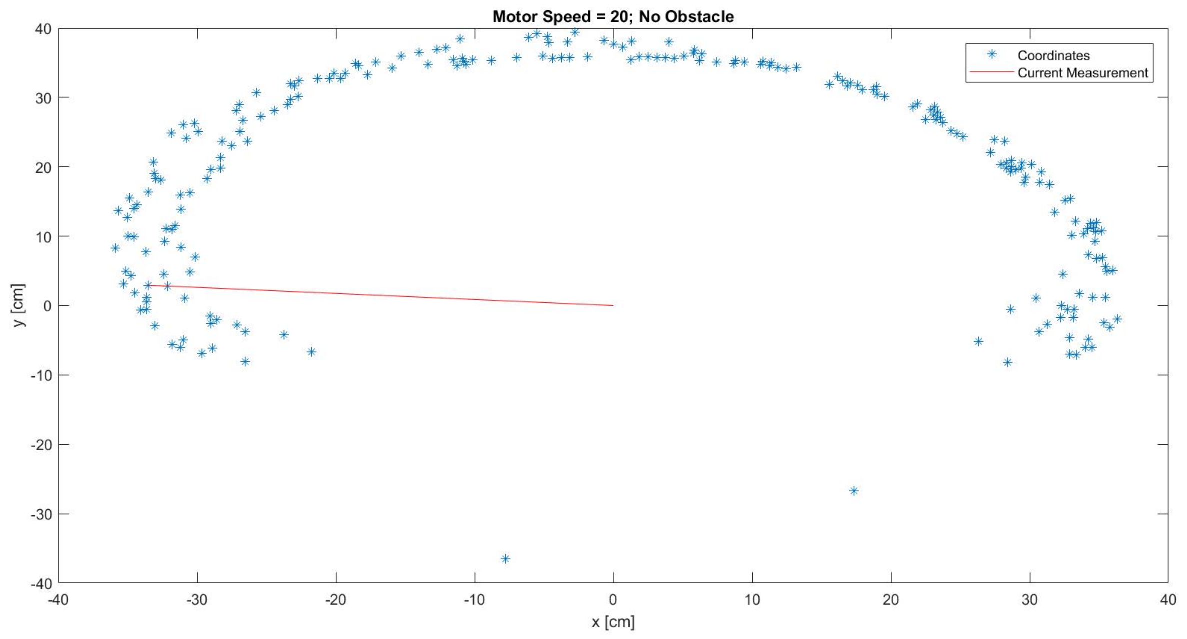
  Note: the coefficient is also reflected as follows:

  ```
  x_array(i) = d_array(i)*cosd(theta)*coefficient;
  y_array(i) = d_array(i)*sind(theta)*coefficient;
  ```
- Towards the end of each loop, the x-y coordinates are plotted.
- Some inbuilt commands like ‘clf’ and pause are used for proper plotting.

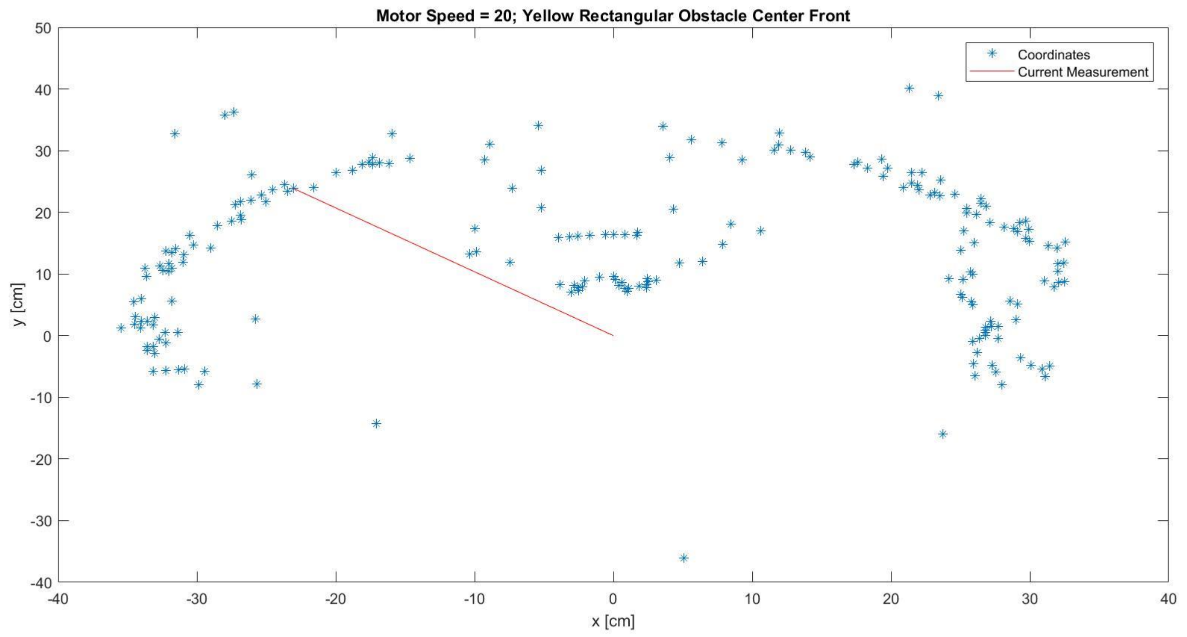
RESULTS

The following results were obtained from the implementation of the MATLAB code on the range finder robot.

Scan Without Obstacles:



Scan With an Obstacle:

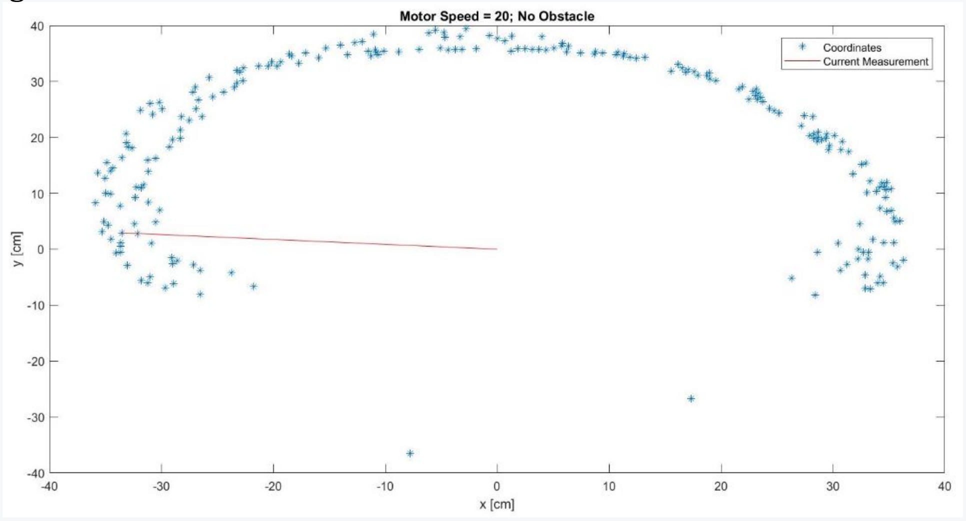


P.S. Extra results are included in the discussion section.

DISCUSSION

Scan Without Obstacles

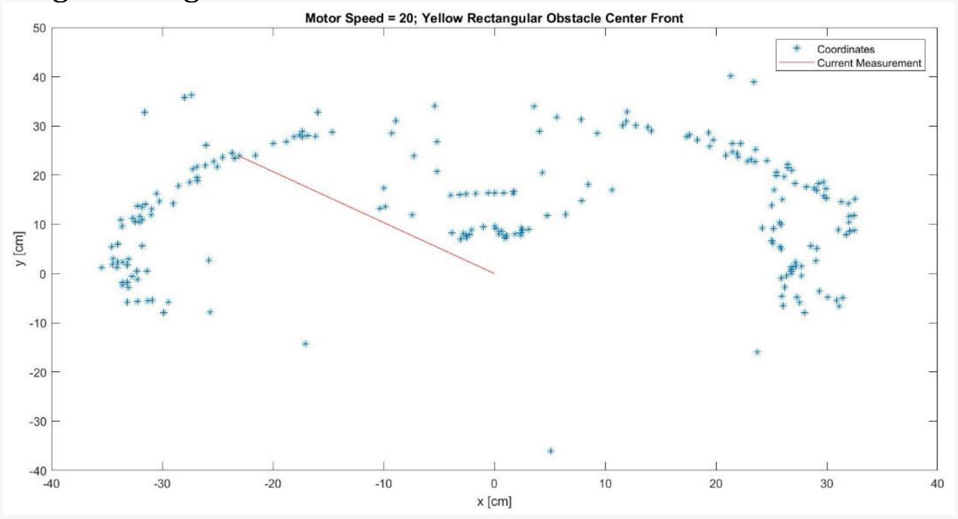
- The resulting path in the scan is a semi-elliptical one. This is as expected because the IR sensor undergoes a semicircular rotation.



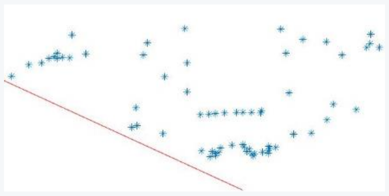
- The measured points are mostly along the elliptical path; however, we notice some slight deviations at either edge (close to 0 degree and close to 180 degrees). **This is due to the impact caused by abrupt stops at both extremes during each scan**. This impact causes the range finder to move slightly away from its original position.
- A solution for this could be to make a setup for continuous 360-degree rotations to avoid the jerk from abrupt stops.

Scan With Obstacles - Q2

- Once an obstacle is added, the resultant plot not longer fully follows an elliptical path. In the ideal case, the detections around the obstacle would sort of map out the shape of the obstacle along the range line.



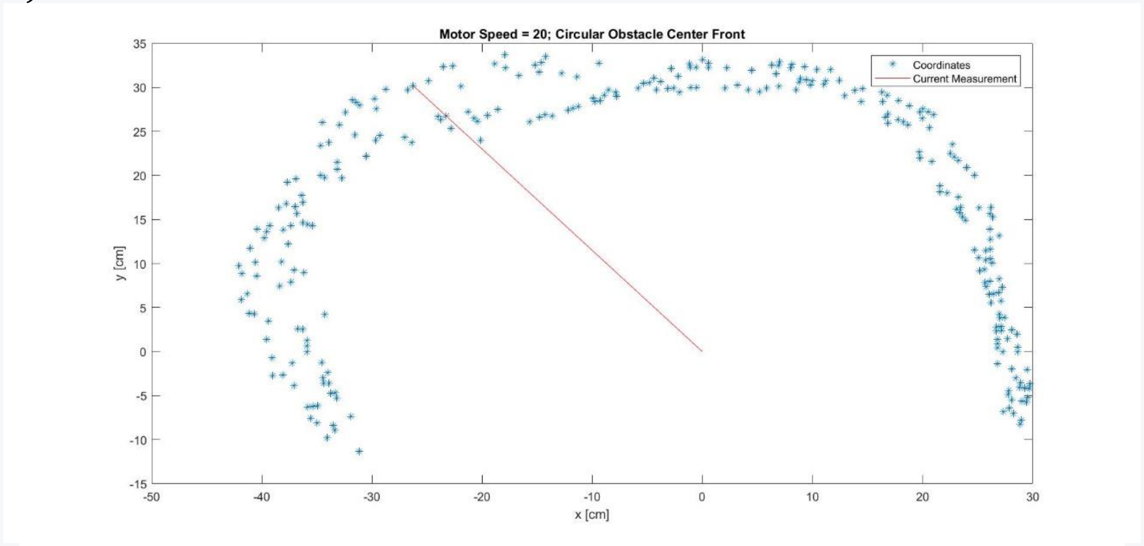
- In the above case, a small rectangular yellow slab was placed in the central position in front of the range finder. As we can see, there are detected points in that region close to the [0,0] point. Also, the points are not exactly along a straight line although the obstacle here is a rectangular one. The reason for this is because as the sensor rotates in a circular path, it is closest the object at 90 degrees (in the perpendicular direction). Away from 90 degrees, the distance gradually decreases; hence, the curved outline as seen in the cropped section below:

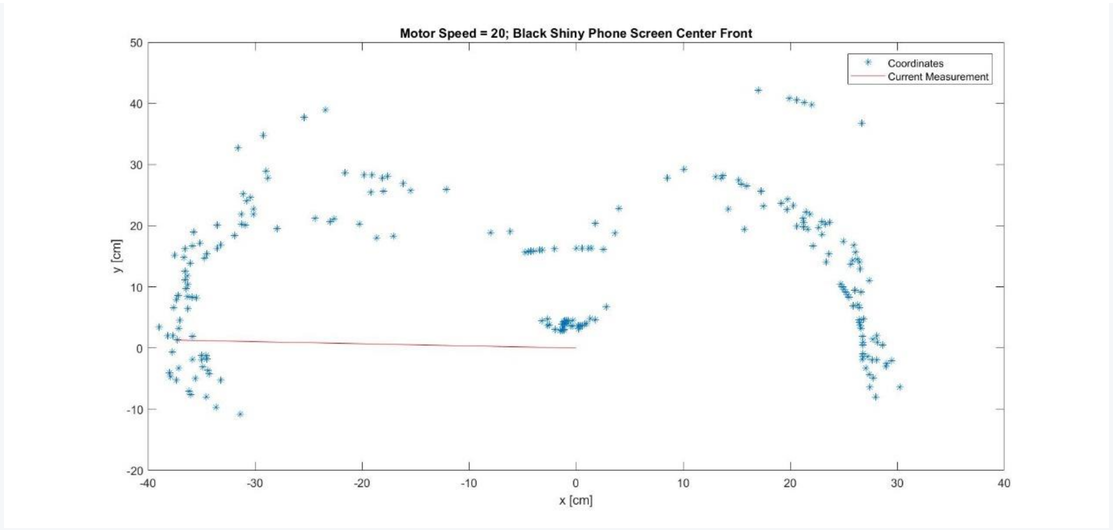


Q3 – Effect of Shape and Color on Measurements

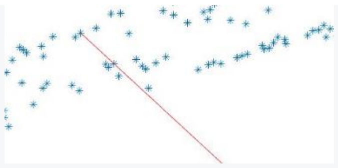
Shape:

- Given the prevalence of systematic and random errors, it is rather difficult to examine the effect of shape on the measurement, but the following was done notwithstanding.
- Firstly, measurements were taken with a robot wheel (with the circumference facing the IR sensor) as an obstacle. Then, the experiment was repeated with a phone screen (rectangular in shape). The results obtained are as follows:





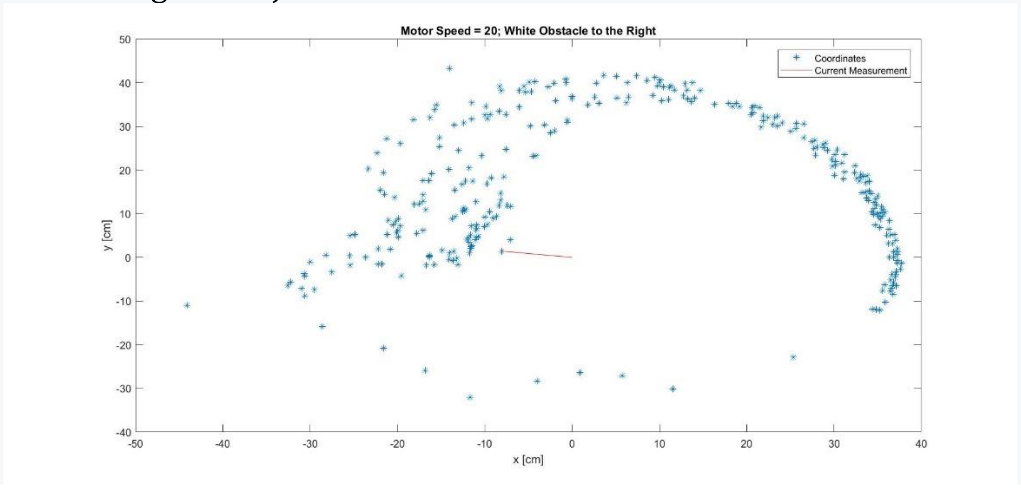
- From the above, in the first image, there's a slight inward curving outline around the region of the red line. That corresponds to the detections around the circular wheel.

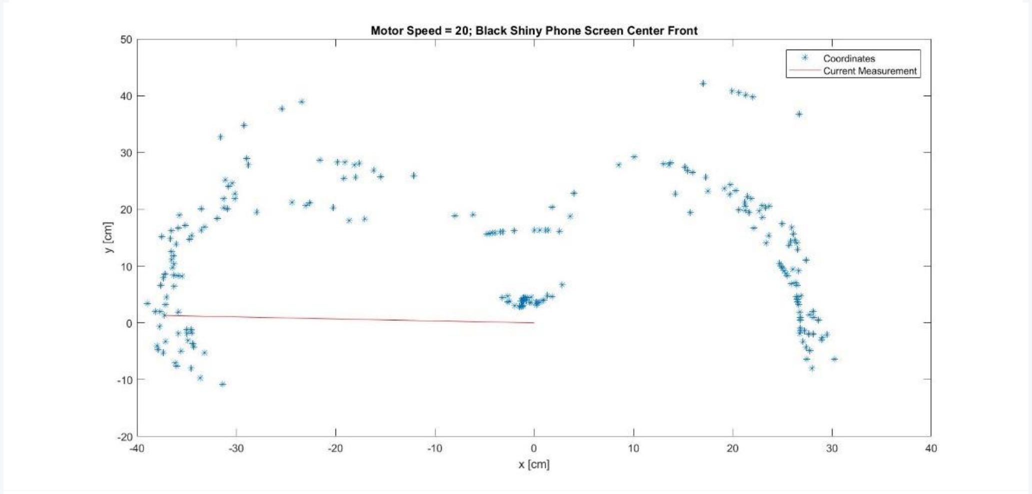


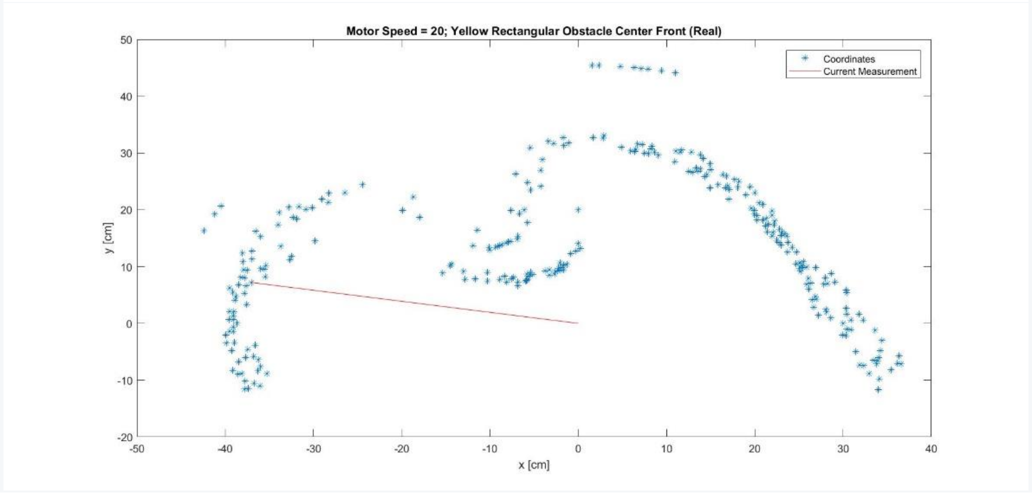
- In the second image for the rectangular phone screen, the outline is 'flatter'.
- Although difficult to identify, flatter surfaces like rectangles produce flatter outlines, while curvy surfaces like wheels produce curvy outlines.

<u>Color</u>:

- The experiment was performed for a black object (a black shiny screen of a phone), a yellow slab, and a white irregular object.



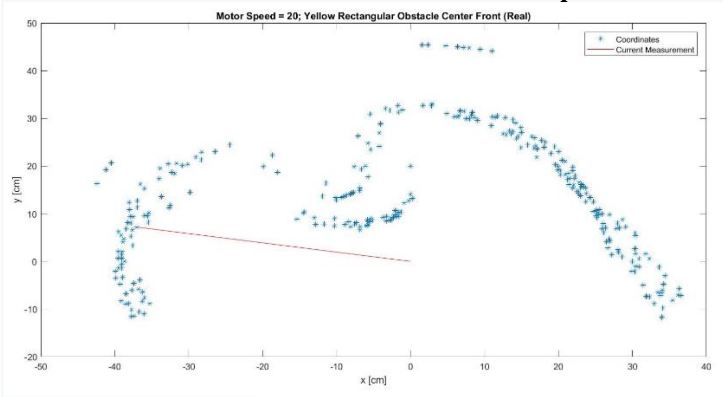


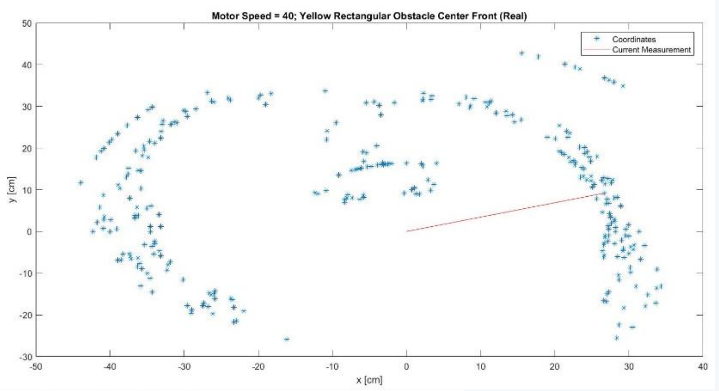


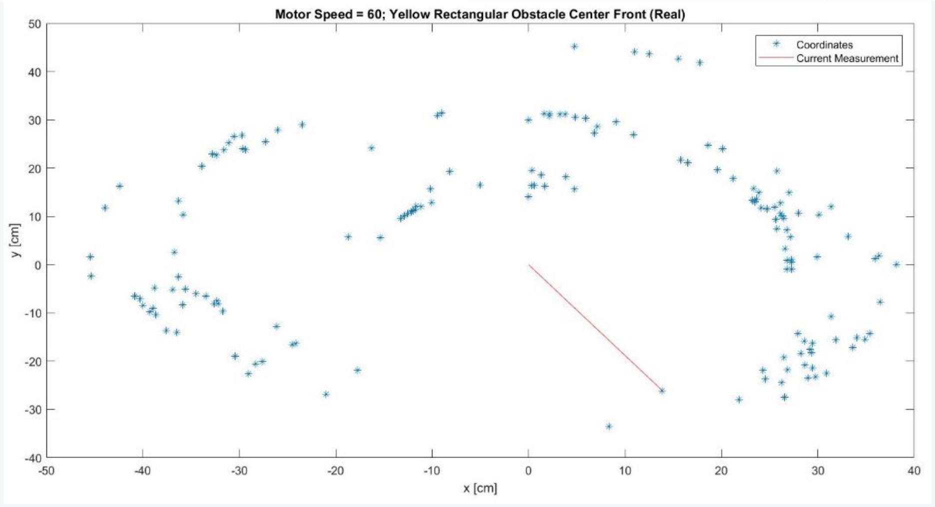
- All objects were placed at approximately the same distance from the sensor.
- From the above results, we observe that the sensor perceives the black object as been really close to it with y-coordinates below 5cm. For the yellow obstacle at the same distance, the sensor perceives it as being farther away compared to the black object with y-coordinates around 8-10cm. For the white obstacle, the results are farther away compared to the black one and a lot like the yellow one.
- **Based on my results, the detections for the black object were closer than the real distance, so I believe that dark objects do not yield good range estimates.**

Q4 – Effects of Speed on Measurements

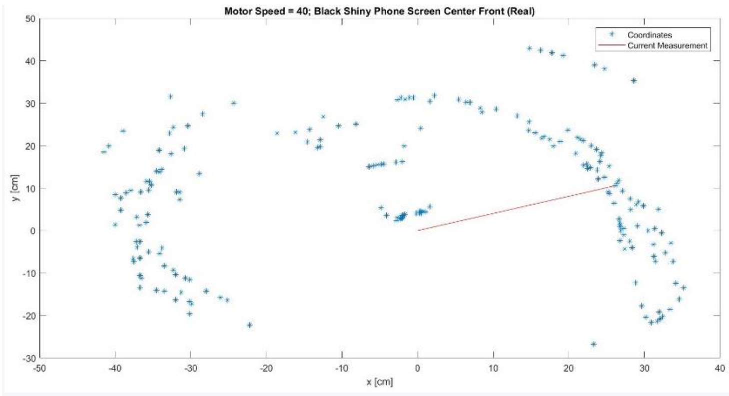
- The speed of the motor directly affects the measurements. This is evident due to the uneven readings produced when the speed of the motor is increased. This is as a result of the inevitable vibration that comes with higher speeds. Also, the higher torque in this case is more detrimental to the abrupt stop and reverse at the two extremes (0 and 180 degree). Observe the results for motor speeds of 20, 40, and 60 below:

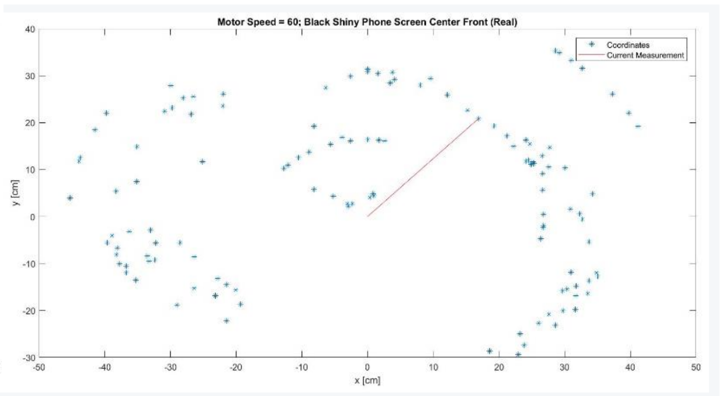




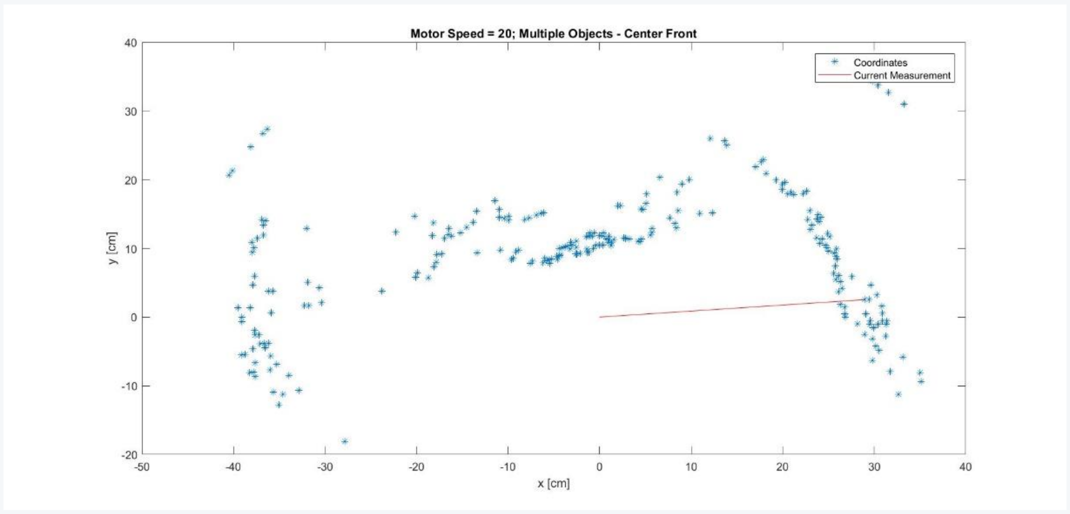


- As we increase the speed, the readings become more spaced out.
- Some worthy of noting is the fact that there are significantly more readings that fall outside the scan range of 0-180. This is due to reasons mentioned in the last few sentences.





Multiple Obstacles



- Three objects were arranged side-by-side in front of the IR sensor, and the above result was obtained.

CONCLUSION

This lab includes a condensed study on the implementation of an active ranging sensor. The sensor in this case is an Infrared sensor. With the sensor attached to a motor for semi-circular ranging, the reflection of the infrared light was used to determine the distance of the obstacles from the sensor. During this experiment, factors affecting the durability of an active ranging device were investigated. These factors include, but are not limited to, shape of object, distance from sensor, color of object, time taken to scan (reflected in the speed of the motor) etc. Furthermore, systematic errors like vibration, errors from stop and reverse motions were encountered. In conclusion, the IR sensor is an **exteroceptive and active sensor**.

APPENDIX

Major Code

```
clear all;
clc;

mylego = legoev3('usb');

% Set up infrared sensor on port 3
myirsensor = irSensor(mylego, 3);

% Set up the motor
MyMotor = motor(mylego,'A');

% Set up the parameters
d_array = [];
x_array = [];
y_array = [];
theta_array = [];

resetRotation(MyMotor);
start(MyMotor);
MyMotor.Speed = 20;

% The coefficient is for conversion is obtained through a
simple test on
% the IR sensor.
coefficient = 30/66;
i = 1;

while ~readButton(mylego, 'up')
    theta = double(readRotation(MyMotor));
    % Conditions to maintain the scanning range between 0
and 180 degrees
    if theta < 0
        MyMotor.Speed = 20;
        start(MyMotor);
    end
    if theta > 180
        MyMotor.Speed = -20;
        start(MyMotor);
    end

    % Calculate the distance & angle, and update the
coordinates.
    theta_array(i) = theta;
    d_array(i) = readProximity(myirsensor);
    x_array(i) = d_array(i)*cosd(theta)*coefficient;
    y_array(i) = d_array(i)*sind(theta)*coefficient;

    % Plot the results
    clf;
    plot(x_array, y_array, '*');
    hold on;
    plot([0 x_array(i)], [0 y_array(i)], '-r')
    xlabel('x [cm]')
    ylabel('y [cm]')
    legend('Coordinates','Current Measurement' )
    pause(0.001)
    i = i+1;
end

stop(MyMotor);
```