Lab Report 5

Group #12

Presented by:
Andréanne Dubuc (260868186)
Aleksas Murauskas (260718389)
Bob Qian (260763075)
Chang Zhou (260779060)
Leonidas Scordos (260799924)
Talaal Mazhar Shafi (26086528)

Design Principles and Methods
ECSE 211

McGill University
February 28th, 2019

Section 1 : Design Evaluation

This portion of the paper will entail of the design of our robot. The hardware design was kept similar to lab 4 because we wanted to reuse the ultrasonic and light localization classes to better navigate the target area. Through testing, the robot's track was determined to be 13.4 cm and the wheel radius was found to be 2.1 cm. Attached to the rear, near the roller ball, is a light sensor to be used for light localization [visible in figure 2]. At the front of the vehicle, an ultrasonic sensor was placed to be used in both ultrasonic localization and the detection of cans. Additionally a gyrometer was added to assist with localization of the odometer's theta value[visible in figure 2]. On the right side of the robot, a large motor was attached with a light sensor adjoined to the rotating axle [visible in figure 1]. This light sensor was placed at a height were most of the cans have a solid strip of their color around 80% of the can. This height was chosen to avoid the light sensor landing on labels or parts of the can that would obscure it's true color.

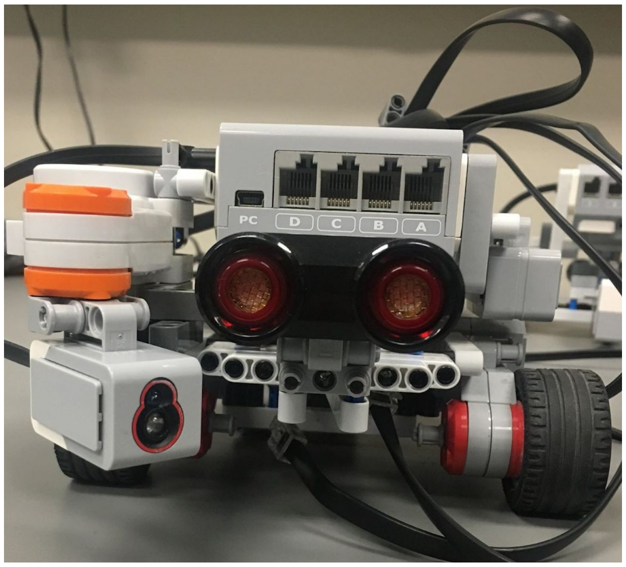
Figure 1: Front view of the robot

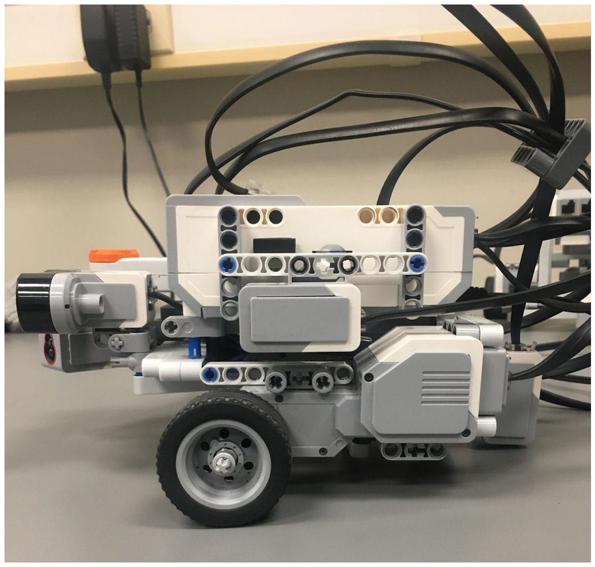
Figure 2: Side view of the robot

Overall design of the software was kept consistent with Lab 4, as the only new Class introduced was ColorClassification, which uses the mounted color sensor to detect the color of a can. Before the code is uploaded to the EV3 brick, several variables must be encoded. The variables LL, UR, and the target color must be filled with the lower left hand corner of the search area, the upper right hand corner of the search area, and the index of the target color respectively. When the program is first booted up on the robot, it prompts the user for the type of operation the user desires, a color detection test or a field test. Pressing left will activate the color detection test and pressing right will begin the field test. The color detection test begins by bringing online the US sensor, which polls the distance to the nearest object. If an object is sufficiently in range of the light sensor, the ColorClassification method findColor() is called. The findColor() method contains a Do Loop which repeats until a color is found or the number of loops have exceeded the maximum detection attempts. Within the loop, the program attempts to match the fetched light sensor data to the known color values. This is accomplished with the matchColor() method. First matchColor() normalized the fetched array creating a square average using the equation:

$$sqrAve = \sqrt{R^2 + G^2 + B^2}\ .$$

This square average is divided from the RGB values which results in a normalized array of Red, Green, and Blue values. matchColor() then rotates through the possible colors the object can be and checks if the array falls within a specified interval of the color. The WithinInterval() method returns true if the the RGB values absolute value of the subtraction of the fetched values from the hardcoded RGB values falls within an acceptable tolerance. If the Withininterval() method returns true, the color being tested at the time is returned. If no match within interval is found, then the matchColor() method returns a -1, signifying the fetched data does not match any of the colors. The Display will then show that an object was detected and the color retrieved from matchColor().

If right is selected then the field test is selected. First the robot will use it's US sensor to localize itself in the corner, using the falling edge system developed in lab 4. After that, the robot will perform a light localization using the code developed in lab 4 as well. After the robot is fully localized and on the origin, the navigation thread is created and the robot travels to the lower left corner of the field. Then the difference between the corner's are calculated along both the x and y axis. If the difference in the x axis is smaller than the difference, then moving vertically will be more efficient to search, and vice versa. Within VerticalPlanning() is where the robot navigates the search field, HorizontalPlanning() operates in an equivalent matter just for the other axis. The firstside boolean records which way the robot is facing, true for 0 degrees, false for 180. The program will follow the path displayed in figure 3 if the length of the search area along the x axis is less than or equal to the length along the y axis. The robot follows this path until the Ultrasonic sensor detects a can, then the robot slowly approach and begins to use the color classification to detect the can's color in the method. The robot will continue following the path until the target can is found, or the whole grid has been searched.

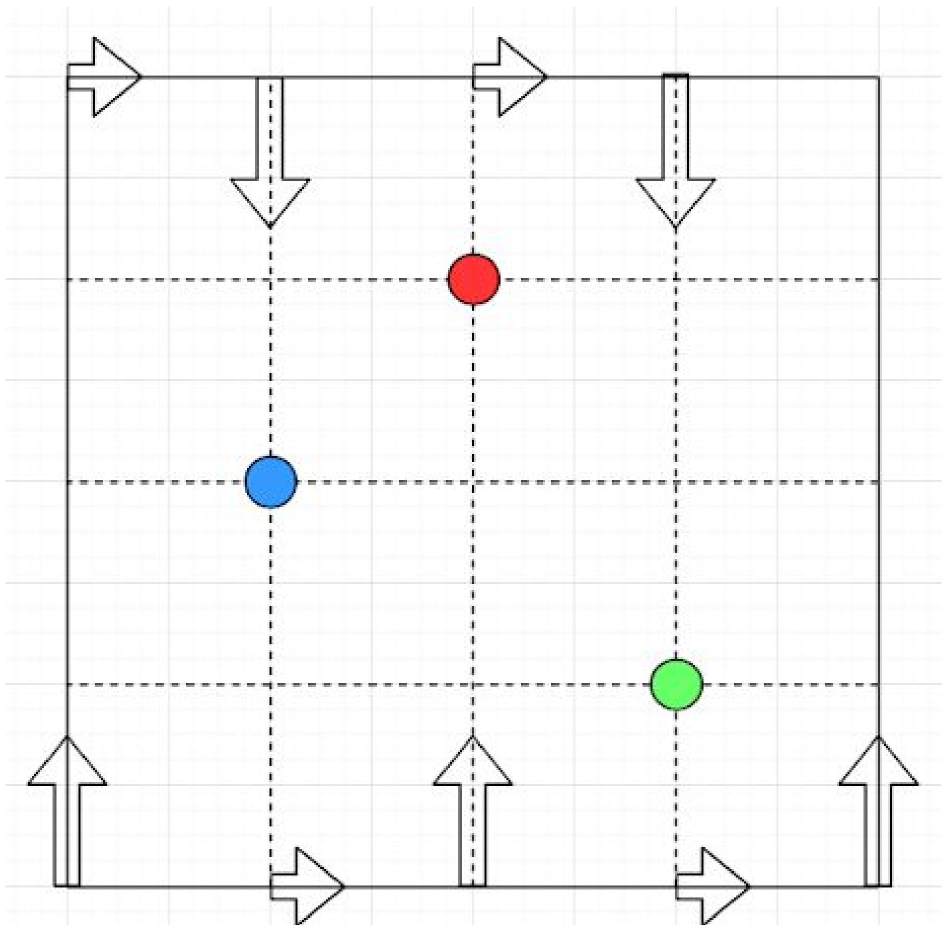

Figure 3: Path

The flowchart and the class diagram found in Figure 4 and Figure 5 give an overview of the software used.

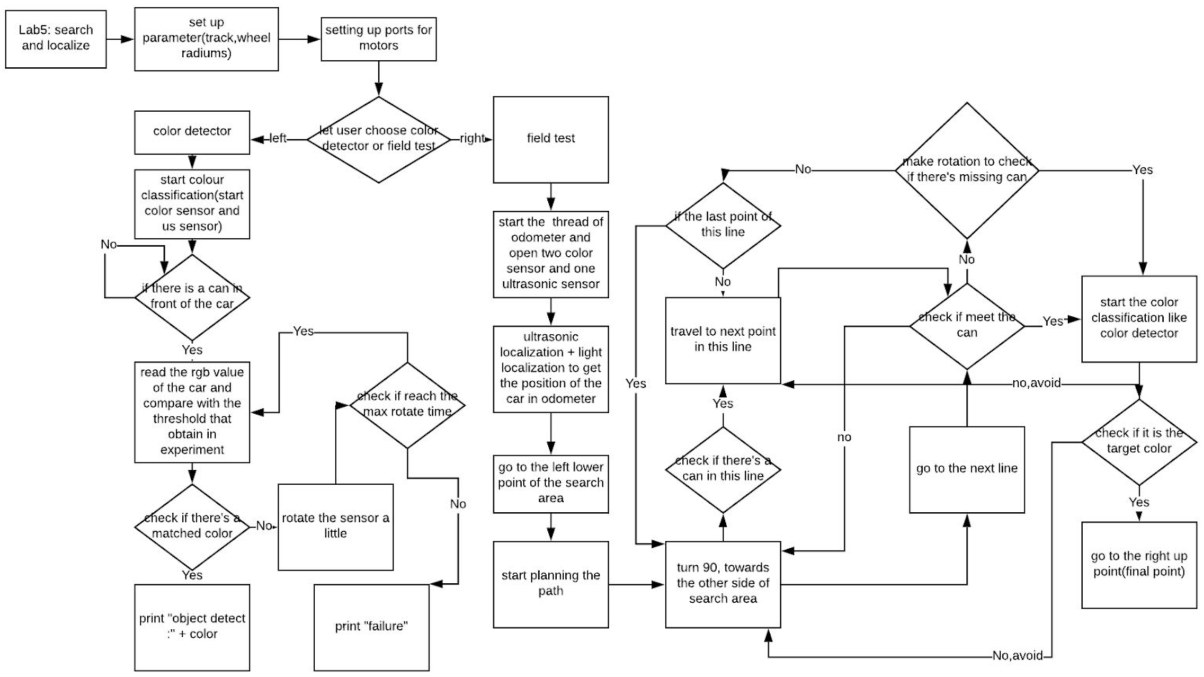
Lab5: search and localize
set up parameter(track,wheel radiums)
setting up ports for motors
let user choose color detector or field test
left
right
color detector
field test
start colour classification(start color sensor and us sensor)
No
if there is a can in front of the car
Yes
read the rgb value of the car and compare with the threshold that obtain in experiment
check if there's a matched color
No
Yes
print "object detect :" + color
rotate the sensor a little
check if reach the max rotate time
Yes
No
print "failure"
start the thread of odometer and open two color sensor and one ultrasonic sensor
ultrasonic localization + light localization to get the position of the car in odometer
go to the left lower point of the search area
start planning the path
turn 90, towards the other side of search area
check if there's a can in this line
Yes
travel to next point in this line
if the last point of this line
No
Yes
No
make rotation to check if there's missing can
Yes
No
check if meet the can
Yes
no
go to the next line
start the color classification like color detector
no,avoid
check if it is the target color
Yes
No,avoid
go to the right up point(final point)

Figure 4: Flowchart of the software

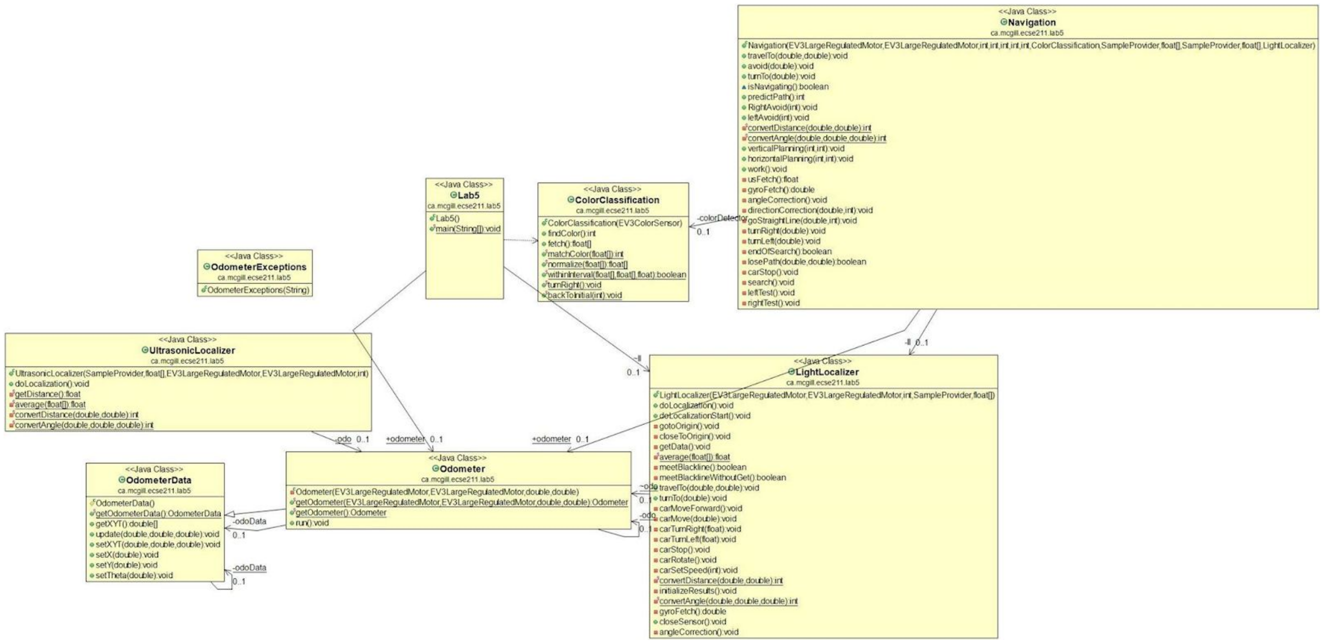

Figure 5: Class diagram of the software

Section 2 : Test Data

Model Acquisition

The following table describes the RGB color values for the Blue can. 10 trials were carried out. Every measurement was taken in the working range of the sensor which has a can-to-sensor distance of 3 to 4 cm.

Table 1: The RGB color values for the Blue can detected by the light sensor

Trials	R	G	B
1	0.039215688	0.105882354	0.087254904
2	0.04019608	0.10784314	0.0882353
3	0.04019608	0.10980392	0.0882353
4	0.039215688	0.10784314	0.08627451
5	0.04019608	0.106862746	0.087254904
6	0.04019608	0.10784314	0.0882353
7	0.0401960	0.105882354	0.087254904
8	0.04019608	0.10882353	0.08921569
9	0.04019608	0.10882353	0.0882353
10	0.039215688	0.106862746	0.08921569

The following table describes the RGB color values for the Green can. 10 trials were carried out. Every measurement was taken in the working range of the sensor which has a can-to-sensor distance of 3 to 4 cm.

Table 2: The RGB color values for the Green can detected by the light sensor

Trials	R	G	B
1	0.023529412	0.09803922	0.021568628
2	0.024509804	0.09803922	0.021568628
3	0.024509804	0.09901961	0.02254902

4	0.023529412	0.097058825	0.020588236
5	0.023529412	0.09901961	0.020588236
6	0.023529412	0.097058825	0.021568628
7	0.024509804	0.09803922	0.020588236
8	0.023529412	0.09803922	0.021568628
9	0.024509804	0.09803922	0.02254902
10	0.024509804	0.09901961	0.02254902

The following table describes the RGB color values for the Red can. 10 trials were carried out. Every measurement was taken in the working range of the sensor which has a can-to-sensor distance of 3 to 4 cm.

Table 3: The RGB color values for the Red can detected by the light sensor

Trials	R	G	B
1	0.13431373	0.030392157	0.015686275
2	0.13529412	0.03137255	0.016666668
3	0.13431373	0.029411765	0.016666668
4	0.13627452	0.029411765	0.016666668
5	0.13431373	0.028431373	0.015686275
6	0.13529412	0.030392157	0.01764706
7	0.13431373	0.029411765	0.015686275
8	0.13529412	0.030392157	0.015686275
9	0.13431373	0.030392157	0.016666668
10	0.13529412	0.030392157	0.018627452

The following table describes the RGB color values for the Yellow can. 10 trials were carried out. Every measurement was taken in the working range of the sensor which has a can-to-sensor distance of 3 to 4 cm.

Table 4: The RGB color values for the Yellow can detected by the light sensor

Trials	R	G	B
1	0.11764706	0.06764706	0.016666668
2	0.11666667	0.06764706	0.015686275
3	0.11862745	0.07058824	0.01764706
4	0.11764706	0.06764706	0.016666668
5	0.11764706	0.068627454	0.016666668
6	0.11862745	0.06960785	0.01764706
7	0.11666667	0.068627454	0.01764706
8	0.11666667	0.06764706	0.016666668
9	0.11862745	0.068627454	0.01764706
10	0.11862745	0.06764706	0.016666668

Color and Position Identification

The following table represents the RGB values for the target and non-target cans, the closest grid intersection to the Red target can as the real position when the (LLx,LLy), (URx,URy) and SC are respectively (3,3), (7,7) and 0.

Table 5: RGB values for each can, closest grid intersection and real position of the Red target can

Target can: Red					
Can evaluated with their respective color	R	G	B	Closest grid intersection (TPEx, TPEy) to the target (in tile units)	Real position (TPRx, TPRy) (in tile units)
Red can	0.9437581	0.1927922	0.1434182	(6,4)	(6,4)

Blue can	0.3333333	0.6666666	0.6666666	---	---
Yellow can	0.8799064	0.5134091	0.11345761	---	---
Green can	0.2361088	0.9218355	0.1976258	---	---

The following table represents the RGB values for the target and non-target cans, the closest grid intersection to the Blue target can as the real position when the (LLx,LLy), (URx,URy) and SC are respectively (3,3), (7,7) and 0.

Table 6: RGB values for each can, closest grid intersection and real position of the Blue target can

Target can: Blue					
Can evaluated with their respective color	R	G	B	Closest grid intersection (TPEx, TPEy) to the target (in tile units)	Real position (TPRx, TPRy) (in tile units)
Red can	0.8728716	0.4364358	0.2182179	---	---
Blue can	0.3333333	0.6666666	0.6666666	(4,6)	(4,6)
Yellow can	0.8451542	0.50709254	0.16903085	---	---
Green can	0.2649065	0.92717266	0.2649065	---	---

The following table represents the RGB values for the target and non-target cans, the closest grid intersection to the Green target can as the real position when the (LLx,LLy), (URx,URy) and SC are respectively (3,3), (7,7) and 0.

Table 7: RGB values for each can, closest grid intersection and real position of the Green target can

Target can: Green					
Can evaluated with their respective color	R	G	B	Closest grid intersection (TPEx, TPEy) to the target (in tile units)	Real position (TPRx, TPRy) (in tile units)
Red can	0.97332853	0.16222143	0.16222143	---	---
Blue can	0.26534325	0.72969395	0.6301902	---	---

Yellow can	0.9023973	0.4153892	0.11459014	---	---
Green can	0.2657065	0.93327266	0.2655065	(5,6)	(5,6)

The following table represents the RGB values for the target and non-target cans, the closest grid intersection to the Yellow target can as the real position when the (LLx,LLy), (URx,URy) and SC are respectively (3,3), (7,7) and 0.

Table 8: RGB values for each can, closest grid intersection and real position of the Yellow target can

Target can: Green					
Can evaluated with their respective color	R	G	B	Closest grid intersection (TPEx, TPEy) to the target (in tile units)	Real position (TPRx, TPRy) (in tile units)
Red can	0.9522657	0.2408972	0.1079231	---	---
Blue can	0.2773524	0.7309721	0.6113458	---	---
Yellow can	0.8439948	0.4991634	0.1257890	(7,3)	(7,3)
Green can	0.2177356	0.9723195	0.1876358	---	---

Section 3 : Test Analysis

Model Acquisition

The following table depicts the mean and the standard deviation for the RGB values for the Blue can.

Table 9: Mean and standard deviation for the RGB values of the Blue can

	Blue Can		
	R	G	B
Mean	0.03990195	0.10764706	0.08794118
Standard deviation	0.00047357	0.00129075	0.00093008

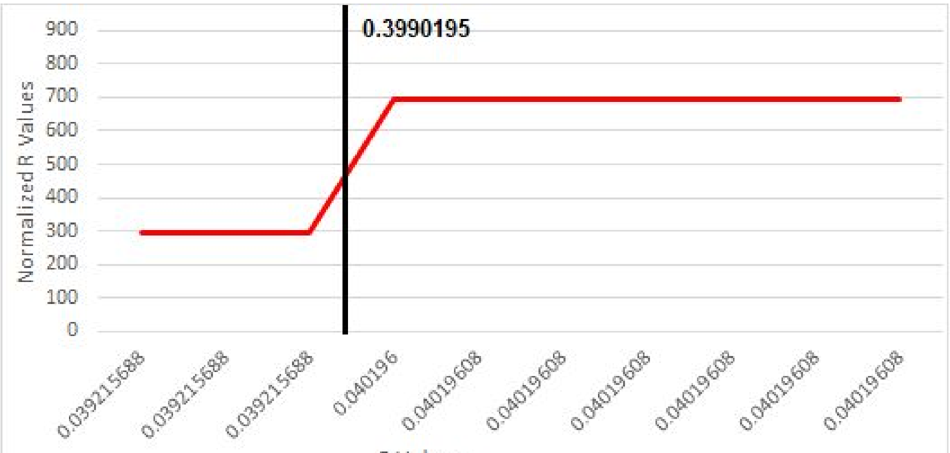

Figure 6: Normalized value of the Blue can for R values

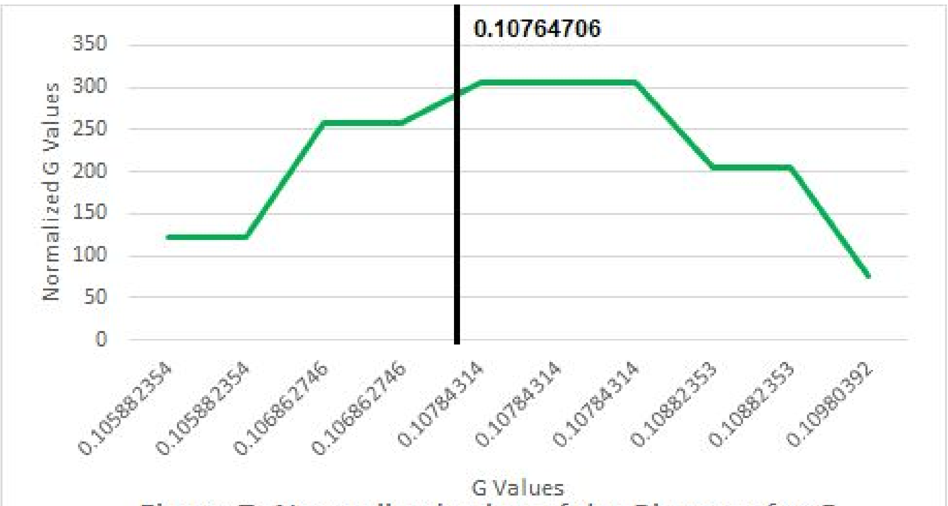

Figure 7: Normalized value of the Blue can for G values

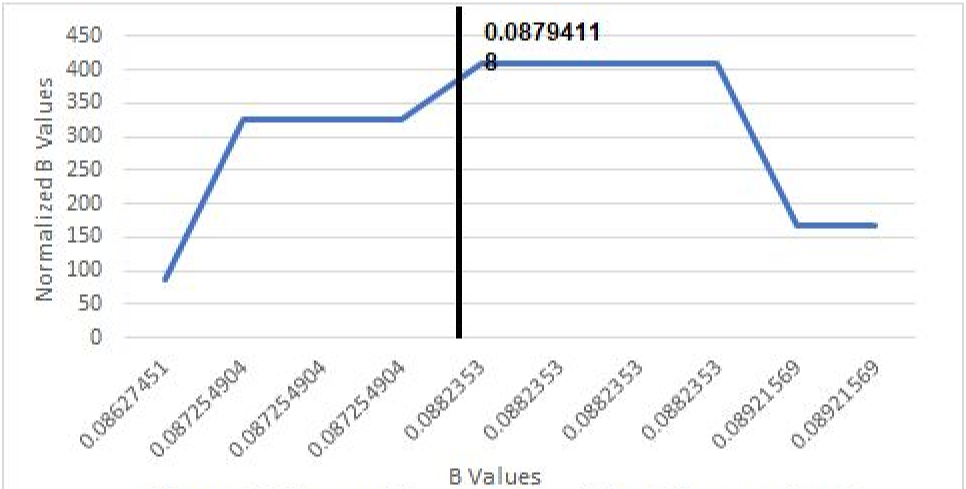

Figure 8: Normalized value of the Blue can for B values

The following table depicts the mean and the standard deviation for the RGB values for the Green can.

Table 10: Mean and standard deviation for the RGB values of the Green can

	Green Can		
	R	G	B
Mean	0.024019608	0.09813726	0.021568628
Standard deviation	0.000516712	0.0007234	0.000800487

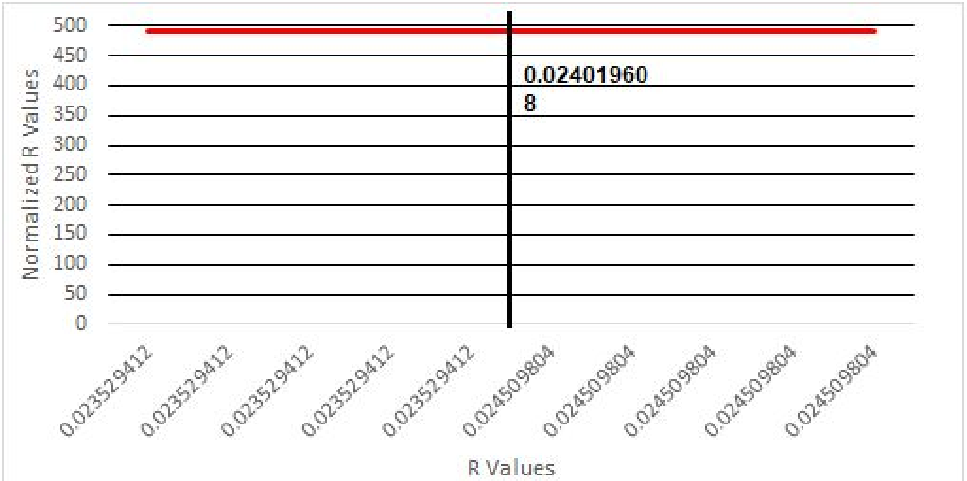

Figure 9: Normalized value of the Green can for R values

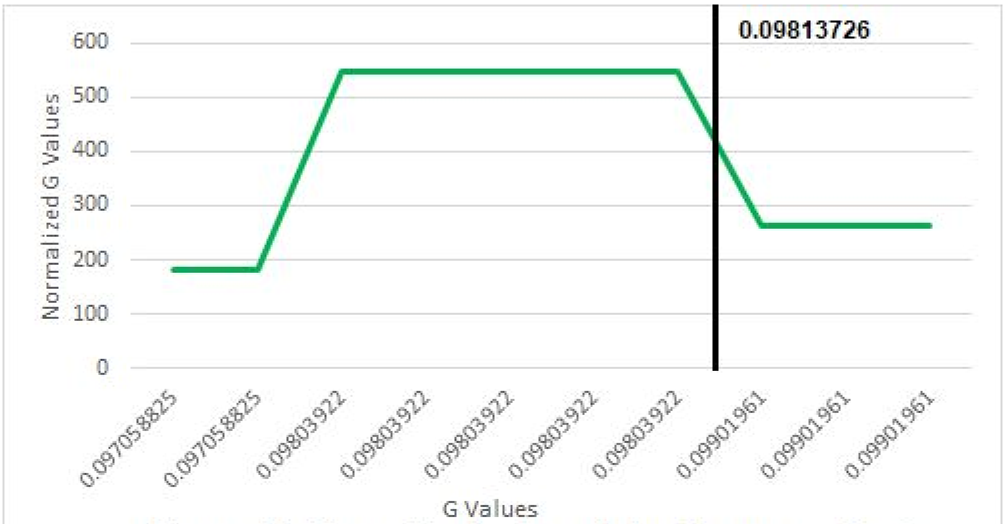

Figure 10: Normalized value of the Green can for G values

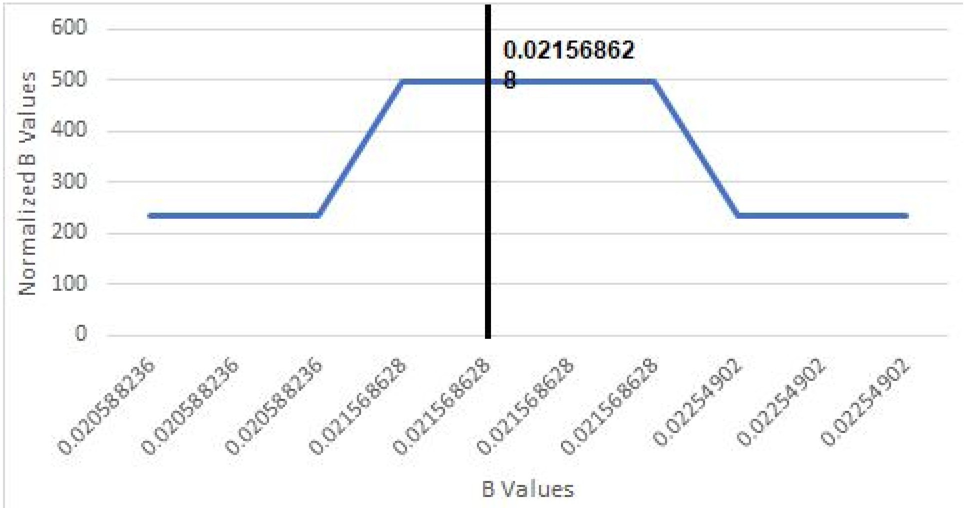

Figure 11: Normalized value of the Green can for B values

The following table depicts the mean and the standard deviation for the RGB values for the Red can.

Table 11: Mean and standard deviation for the RGB values of the Red can

	Red Can		
	R	G	B
Mean	0.13490197	0.03000000	0.01656863

Standard deviation	0.000685497	0.00082674	0.000974931

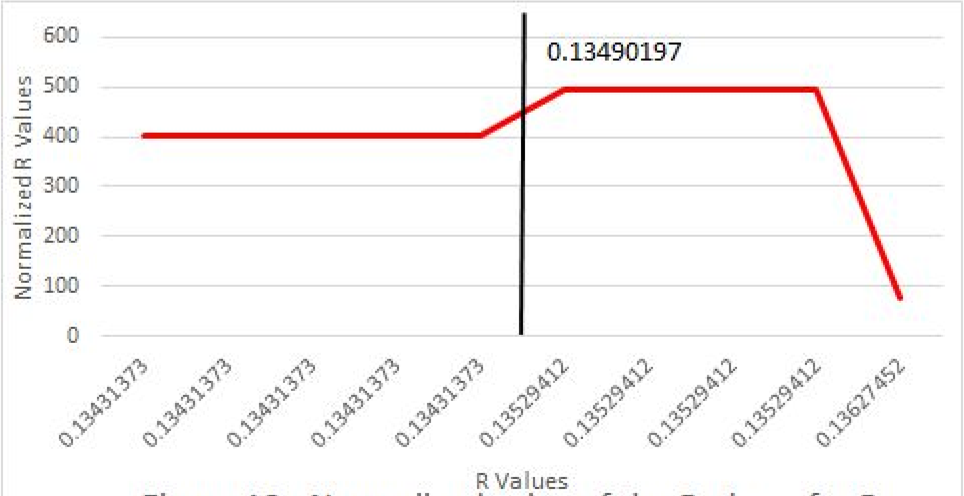

Figure 12 : Normalized value of the Red can for R values

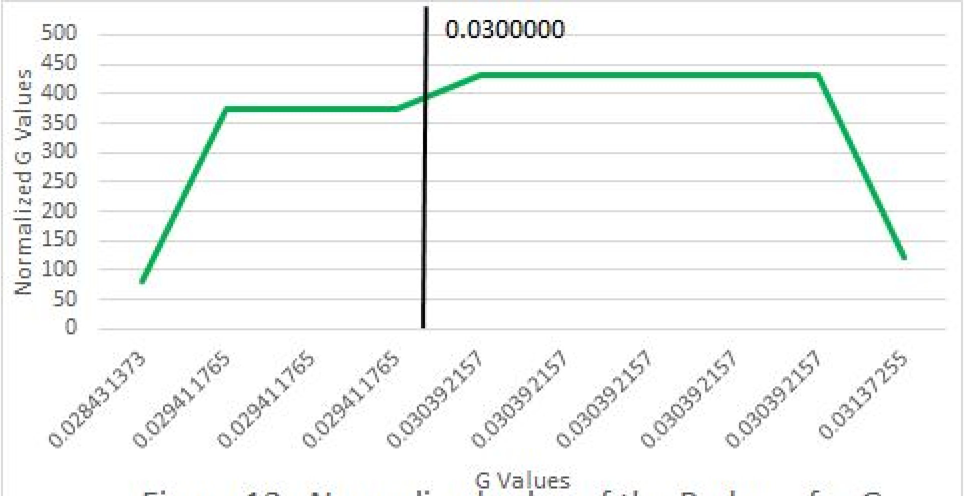

Figure 13 : Normalized value of the Red can for G values

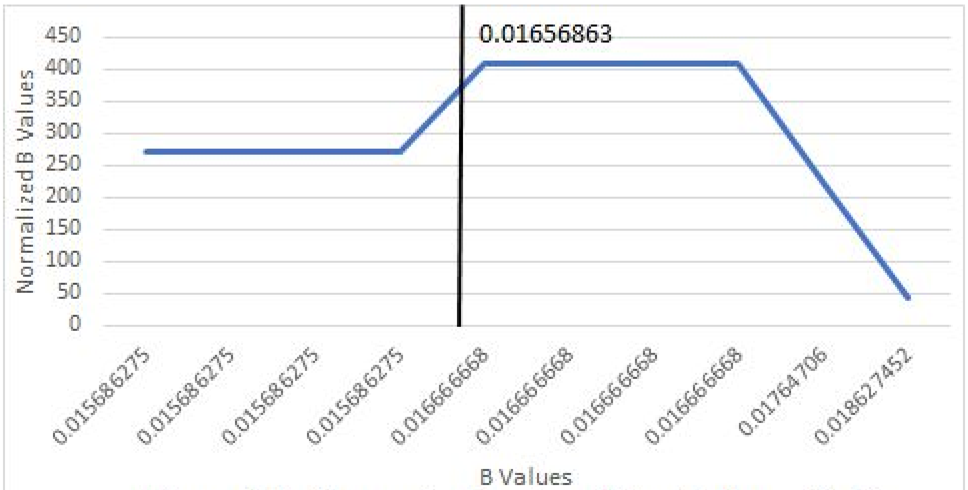

Figure 14 : Normalized value of the Red can for B values

The following table depicts the mean and the standard deviation for the RGB values for the Yellow can.

Table 12: Mean and standard deviation for the RGB values of the Yellow can

	Yellow Can		
	R	G	B
Mean	0.117745099	0.06843138	0.016960786
Standard deviation	0.000858425	0.00101255	0.000661714

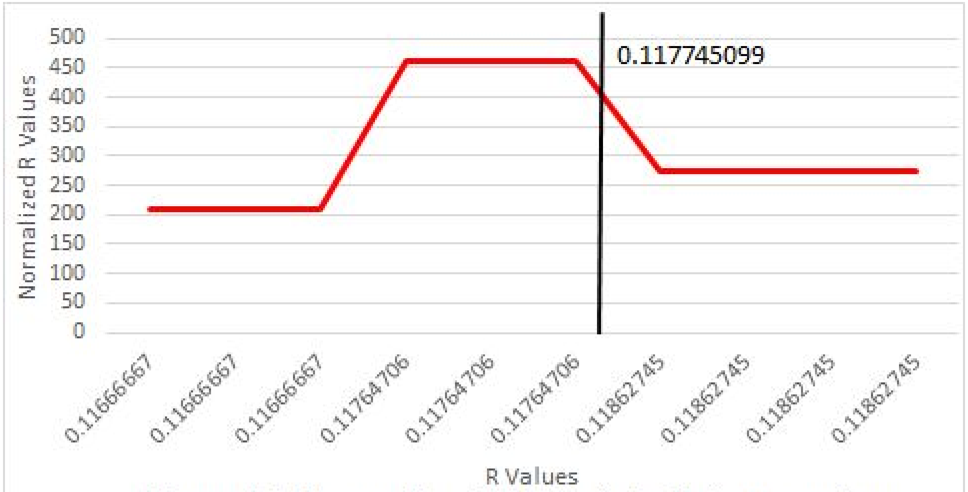

Figure 15: Normalized value of the Yellow can for R values

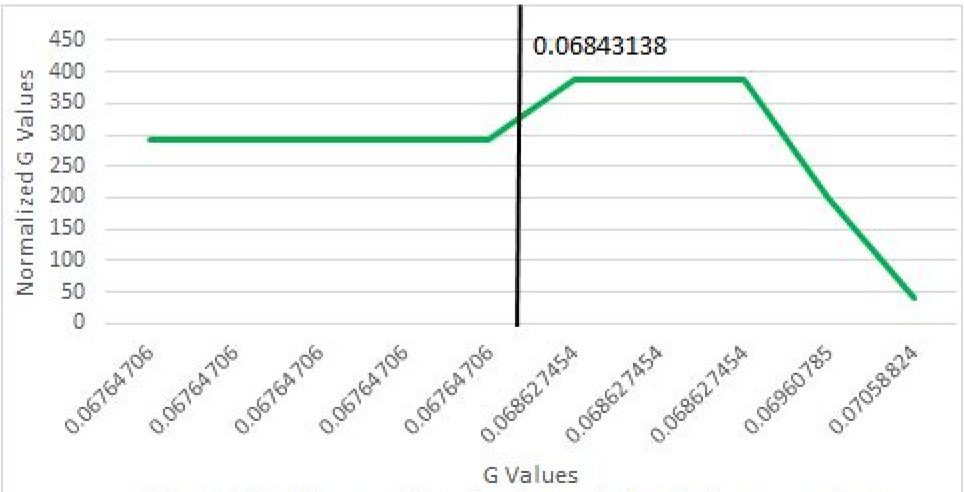

Figure 16: Normalized value of the Yellow can for G values

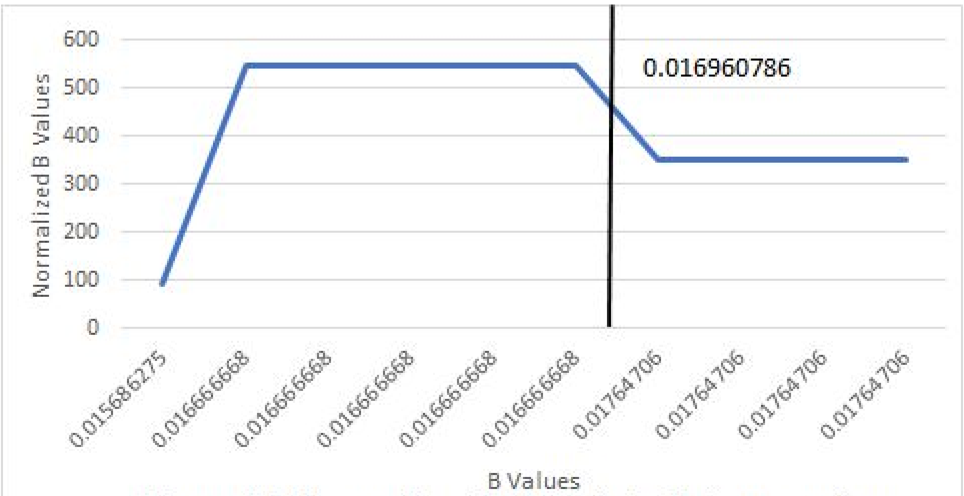

Figure 17: Normalized value of the Yellow can for B values

Sample calculations

To calculate the mean of the RGB values, the formula represented in Figure 18 was used (Khan Academy, 2019). A sample calculation was made using the R values of the Blue can for each trial. The data can be observed in Table 1.

$$\overline{x} = \frac{\sum_a x_i}{n}$$

$$\overline{x} = \frac{0.0392157+0.0401961+0.0401961+0.0392157+0.0401961+0.0401961+0.0401960+0.0401961+0.0401961+0.0392157}{10}$$

$$\overline{x} = 0.03990195$$

Figure 18: Mean formula and a sample calculation

To calculate the sample standard deviation for the RGB values, the formula represented in Figure 19 was used (Khan Academy, 2019). A sample calculation was made using the R values for each trial from Table 1 and its mean value from Table 9.

$$s_k = \sqrt{\frac{\sum_a (x_i - \bar{x})^2}{n-1}}$$

$$s_k = \sqrt{\frac{(0.0392157-0.03990195)^2+(0.0401961-0.03990195)^2+(0.0401961-0.03990195)^2+(0.0392157-0.03990195)^2+(0.0401961-0.03990195)^2+(0.0401961-0.03990195)^2+(0.0401960-0.03990195)^2+(0.0401961-0.03990195)^2+(0.0401961-0.03990195)^2+(0.0392157-0.03990195)^2}{10-1}}$$

$$s_k = 0.00047357$$

Figure 19: Sample standard deviation and a sample calculation

Color and Position Identification

The following table depicts the euclidean distance of the RGB values, the rank and the color detected for every can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) when the target can is Red. To determine the can position (TPEx, TPEy), the values from the odometer class were used.

Table 13: Euclidean distance of RGB values, rank, color detected for each can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) for the target can

Target can: Red				
	Euclidean distance of RGB Values	Rank	Color detected	Euclidean distance between (TPEx, TPEy) and (TPRx, TPRy)
Red can	0.83476965	1	Red	0
Blue can	0.9313131452	4	Blue	---
Yellow can	0.89336635	2	Yellow	---
Green can	0.91563917	3	Green	---

The following table depicts the euclidean distance of the RGB values, the rank and the color detected for every can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) when the target can is Blue. To determine the can position (TPEx, TPEy), the values from the odometer class were used.

Table 14: Euclidean distance of RGB values, rank, color detected for each can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) for the target can

Target can: Blue

	Euclidean distance of RGB Values	Rank	Color detected	Euclidean distance between (TPEx, TPEy) and (TPRx, TPRy)
Red can	0.90493784	4	Yellow	---
Blue can	0.85646244	1	Blue	0
Yellow can	0.90253166	3	Yellow	---
Green can	0.86808178	2	Green	---

The following table depicts the euclidean distance of the RGB values, the rank and the color detected for every can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) when the target can is Green. To determine the can position (TPEx, TPEy), the values from the odometer class were used.

Table 15: Euclidean distance of RGB values, rank, color detected for each can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) for the target can

Target can: Green				
	Euclidean distance of RGB Values	Rank	Color detected	Euclidean distance between (TPEx, TPEy) and (TPRx, TPRy)
Red can	0.96180945	4	Red	---
Blue can	0.90968191	2	Blue	---
Yellow can	0.93853565	3	Yellow	---
Green can	0.90297806	1	Green	0

The following table depicts the euclidean distance of the RGB values, the rank and the color detected for every can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) when the target can is Yellow. To determine the can position (TPEx, TPEy), the values from the odometer class were used.

Table 16: Euclidean distance of RGB values, rank, color detected for each can and the euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) for the target can

Target can: Yellow				
	Euclidean distance of RGB Values	Rank	Color detected	Euclidean distance between (TPEx, TPEy) and (TPRx, TPRy)
Red can	0.85699664	2	Red	---
Blue can	0.90428328	3	Blue	---
Yellow can	0.85135908	1	Yellow	0
Green can	0.92527931	4	Green	---

Sample calculations

To calculate the Euclidean distance of RGB values the formula represented in Figure 20 was used. A sample calculation was made using the mean of the RGB values of the Red can, which was the target can, in Table 11 and the RGB values collected for the Blue can in Table 5. The data can be observed in Table 13.

$$d = \sqrt{(s_R - \mu_R)^2 + (s_G - \mu_G)^2 + (s_B - \mu_B)^2}$$

$$d = \sqrt{(0.3333333 - 0.13490197)^2 + (0.6666666 - 0.0300000)^2 + (0.6666666 - 0.01656863)^2}$$

$$d = 0.9313131452$$

Figure 20: Euclidean distance of RGB values and a sample calculation

To calculate the Euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) the formula represented in Figure 21 was used. A sample calculation was made using the (TPEx, TPEy) values of the Red can and the (TPRx, TPRy) values of the Red can found in Table 5. The data can be observed in Table 13.

$$\epsilon = \sqrt{(TPE_x - TPR_x)^2 + (TPE_y - TPR_y)^2}$$

$$\epsilon = \sqrt{(6-6)^2 + (4-4)^2}$$

$$\epsilon = 0$$

Figure 21: Euclidean distance between (TPEx, TPEy) and (TPRx, TPRy) and a sample calculation

Section 4 : Observations and Conclusions

Rank ordered Euclidean distances are indeed sufficient at determining can colors. As can be shown in figures 13 to 16, the can with the highest rank was the desired can. This is because the distance is adjusted to that color's mean, which will improve the odds that the color will recognized as the target color will have a lower euclidean distance.

The standard deviation of is useful for detecting false positive is useful as when a datapoint falls outside the first standard deviation, it can be filtered out. However, when a large amount of data points fall outside the first standard deviation, then either the can is a different color or a label was crossed.

The color sensor detects color by sending out light and detects the intensity of the light reflected back to the sensor. This allows the color of the can to be determined by what color's the can absorbs. The conditions that work best for detecting colors include the sensor being orthogonal to the can, the sensor being at a height with the most visibility of the can's color. Additionally, a constant ambient light can cause fewer issues to arise when determining the can color.

Section 5 : Further Improvements

Our Software design detects is able to detect the can without rotating the light sensor. Therefore the change in distance created by rotating light sensors is not applied in our model. Using a filter would be helpful to avoid false positives or marking on the can.

To improve the accuracy of your target can's position identification, the ultrasonic sensor could be used. If the sensor was pointing directly at the can, the ultrasonic sensor would give the distance between the sensor and the can. If we know the distance from the robot center of rotation to the can, if we know the radius of the can and if we know the robot's position, all these elements could be combined to know the exact x,y position of the can. Thus, this would give the can a very precise position and would make it easier to grab the can.

Sources

Khan Academy. 2019. 'Mean, median, and mode'. Accessed on 26/01/2019: https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/mean-median-basics/a/mean-median-and-mode-review

Khan Academy. 2019. 'Population and sample standard deviation'. Accessed on 26/01/2019: https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/variance-standard-deviation-sample/a/population-and-sample-standard-deviation-review