ECSE211 DESIGN PRINCIPLES AND METHODS

Lab 5 Search and Localize

Team 17
Nada Marawan
Maxime Cardinal
Irmak Pakis
Ameer Osman
Yi Zhu
Spencer Handfield

1. Design Evaluation

The robot is composed of 2 large motors controlling the wheels and the brick supported just above it so as to minimize its width. To reduce the axis bending caused by the force, the connection axis between both large motors was reinforced and the length of wheel axis on each large motor was reduced. To accomplish the ultrasonic localization which is same as in Lab 4, the ultrasonic sensor is placed at the front bottom robot in a zero-degree angle due to the detection efficiency. We used three color sensors in this lab that one at the front for color of rings detection and two at both sides of the robot for navigation correction. The front light sensor is facing downwards for rings color detection and is held by an arm above the ultrasonic sensor. The distance between color detection and ultrasonic sensor has been tested that the ultrasonic sensor will not detect the light sensor as an obstacle. Other two light sensors are placed right behind these wheels parallelly and at approximately the same height so it can help accomplish both angle and x,y coordinates correction during the navigation. There are a few core components to the software architecture for this iteration.

The overall architecture of the software remains largely unchanged from previous versions with a central Lab 5 class calling other classes which utilize their own methods and the other appropriate classes required for the lab with their own adjusted methods and tested values.

Firstly the localization at the start of the lab was left completely unchanged from prior iterations due to its accuracy and appropriateness for this lab. Second, there is the odometer correction. The odometer in its own is effective and reused, however the correction was modified and improved from previous iterations. It uses the two light sensors on the side of the robot and detects if one crosses before the other, with these readings it is able to stop the wheel as it reaches the line. This process is done for both wheels independently , and thus both wheels, regardless of any prior misalignment along the line, will both stop precisely on such and correct any theta offset while simultaneously correcting the distance it finds itself at (someone error check). And accurately reflecting the location based on the precision of the lines of the grid. This method of correction was chosen as in our trials it demonstrated a high level of accuracy as well as proving itself efficient in its implementation in our code base. Lastly there is the light sensor which detects the color of the rings presented to it. The ultrasonic sensor determines when the robot is appropriately close to the ring. The light sensor information is passed to the appropriate class and tests if its the correct color and beeps accordingly. The robot avoid it, repeating the process with correction until the last corner is reached. The traversal in this manner is undertaken in the search class which requires the coordinates of the area to search and traverses along the lines, calling the correction while continuously polling the US and light sensors in the case of a ring being found. It progresses square by square in the defined region.

The work was undertaken with this final objective and software set up in mind with iterative steps along the way of implementing, testing and adjusting each software component. However while the compatibility between the classes existed, the integration was less than ideal wherein the individuals components functioned successfully but when combined and set in the real lab demonstration failed. They did function in combination to limited enough extent that we were able to gather the data for the following section via the workflow but required further elaboration to reliably complete the entirety of the demonstration

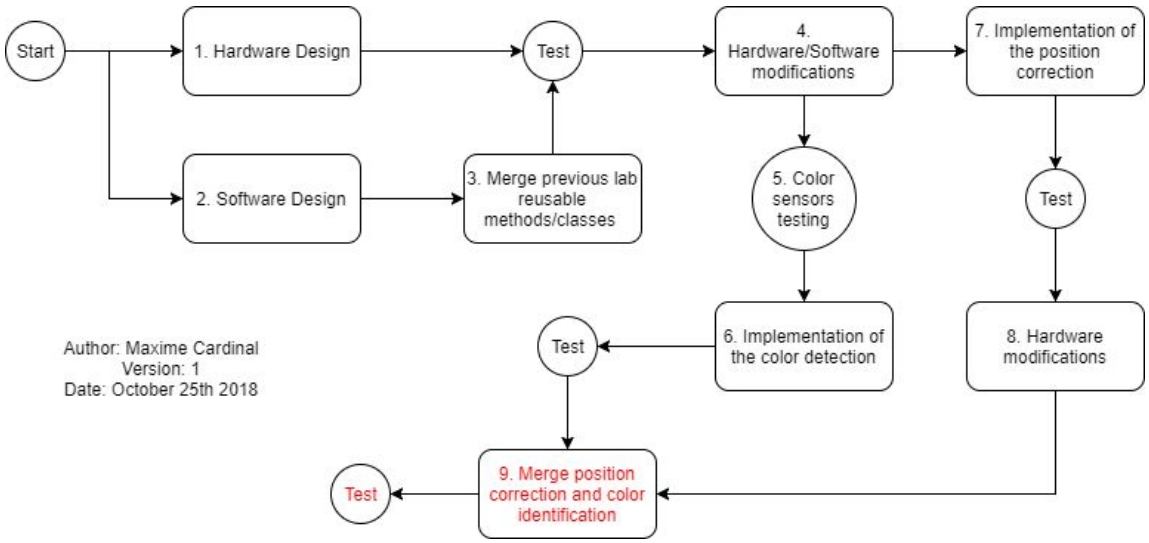


Figure 1 - Workflow

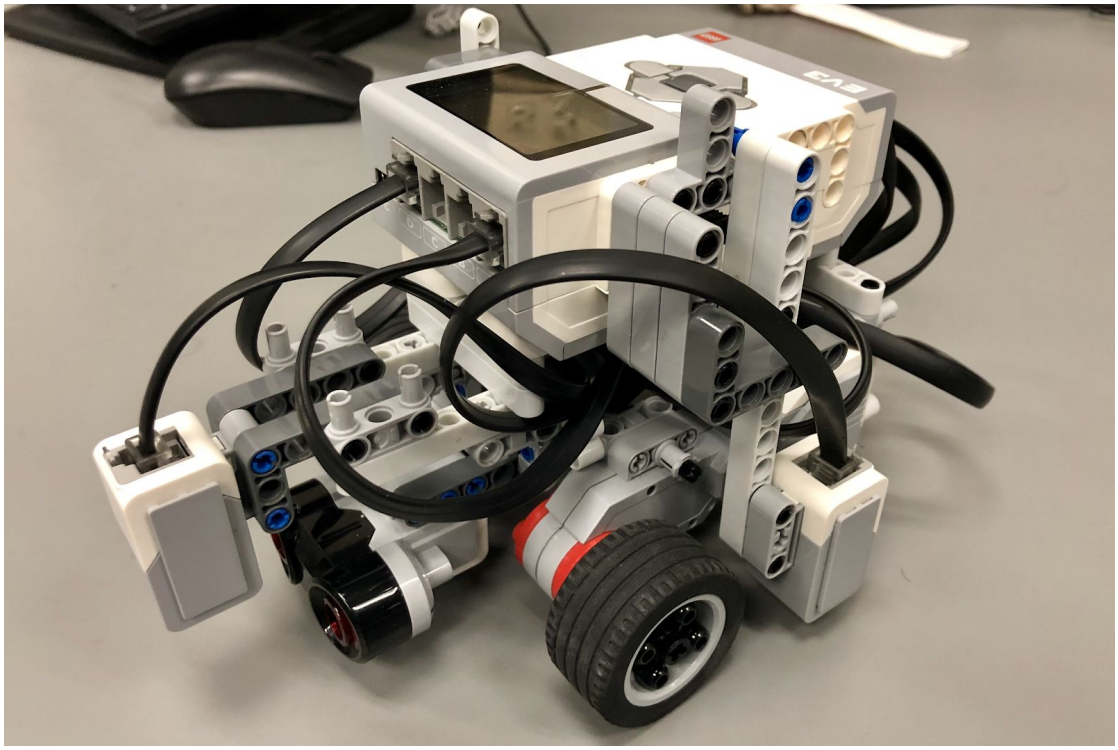


Figure 2 - Hardware design of the robot

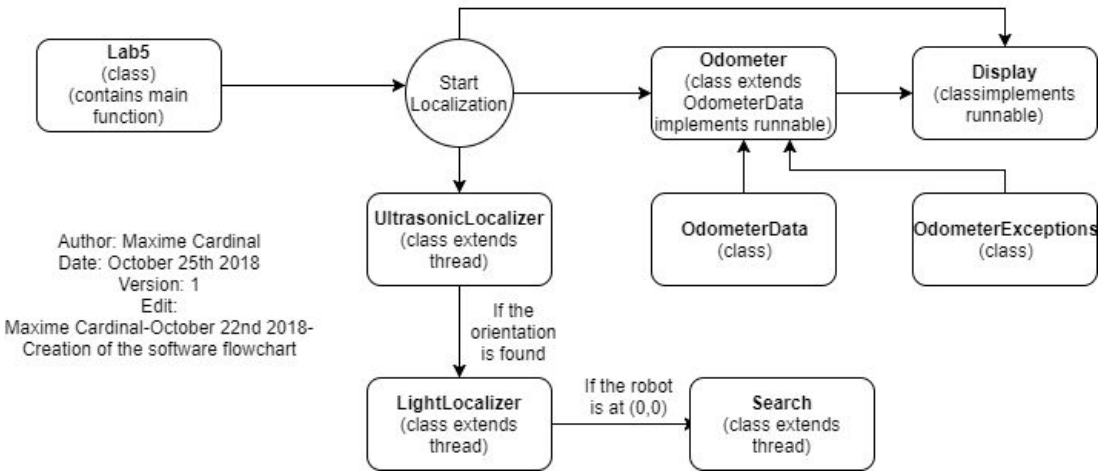


Figure 3 - Software flowchart

2. Test Data

2.1 Model Acquisition

In order to calibrate the color of each target ring, we must place them against a color sensor and read the corresponding RGB values, making sure we read different areas on the surface of the ring. The RGB values data is summarised for each TR in the following tables.

2.1.1 Blue Ring:

Trial #	R	G	B
1	0.021764	0.070196	0.086274
2	0.028647	0.083725	0.092745
3	0.026666	0.082764	0.092764
4	0.028627	0.084705	0.092725
5	0.027647	0.082745	0.091426
6	0.025371	0.083647	0.093192
7	0.026576	0.080664	0.089734
8	0.027382	0.079396	0.090078
9	0.026669	0.082291	0.091062
10	0.027351	0.077537	0.090980

Table 1

2.1.2 Yellow Ring:

Trial #	R	G	B
1	0.218843	0.125294	0.026666
2	0.229607	0.131968	0.032549
3	0.200215	0.114509	0.034705
4	0.223725	0.117254	0.026666
5	0.229607	0.139969	0.029607
6	0.240310	0.120974	0.031947
7	0.229608	0.124661	0.029901
8	0.197801	0.122799	0.034076
9	0.230290	0.119748	0.030291
10	0.203984	0.140694	0.023982

Table 2

2.1.3 Orange Ring:

Trial #	R	G	B
1	0.159803	0.038450	0.013756
2	0.169607	0.041176	0.013734
3	0.142156	0.038411	0.013154
4	0.153941	0.045098	0.014774
5	0.149019	0.043156	0.014725
6	0.150391	0.040961	0.014901
7	0.156097	0.039972	0.014097
8	0.153918	0.042197	0.013948
9	0.154406	0.040098	0.014601
10	0.159712	0.043061	0.013760

Table 3

2.1.4 Green Ring:

Trial #	R	G	B
1	0.072941	0.120392	0.017843
2	0.070980	0.134117	0.021795
3	0.071960	0.135098	0.020773
4	0.070980	0.133137	0.019864
5	0.079803	0.137078	0.021784
6	0.074964	0.130693	0.023681
7	0.073629	0.129816	0.019718
8	0.073833	0.133049	0.020691
9	0.070469	0.134682	0.021879
10	0.073771	0.131578	0.016029

Table 4

2.1.5 Target Ring to Light Sensor Distance table:

	TR to Sensor distance (cm±0.50)									
#Trial:	1	2	3	4	5	6	7	8	9	10
Blue	5.34	5.25	4.97	5.19	5.22	4.99	5.09	5.13	5.44	4.93
Green	5.27	4.82	5.31	5.22	5.37	5.25	5.17	5.12	4.97	5.31
Orange	5.15	5.33	5.19	5.25	4.98	5.06	5.28	5.12	5.21	5.02
Yellow	5.27	5.30	5.05	5.21	5.23	4.95	5.26	5.27	5.13	4.92

Table 5

2.2 Color and Position Identification

To perform Color and Position identification, we set the bounds of our zone containing the rings to (LLx,LLy) = (3 , 3) & (URx,URy) = (7 , 7) & placed target rings of colors Orange, Yellow, Green, and Blue at positions (4,5) ,(6,6), (5,4), & (5,6) respectively. We recorded target ring RGB sample values & the coordinates of the closest grid intersection to the target ring. These values are summarised in the following table.

Trial	Orange Ring					Yellow Ring					Green Ring					Blue Ring				
-	$\$_R$	$\$_G$	$\$_B$	(TPEx, TPEy)	(TPRx,TPRy)	$\$_R$	$\$_G$	$\$_B$	(TPEx,TPEy)	(TPRx,TPRy)	$\$_R$	$\$_G$	$\$_B$	(TPEx,TPEy)	(TPRx, TPRy)	$\$_R$	$\$_G$	$\$_B$	(TPEx, TPEy)	(TPRx, TPRy)
1	0.153728	0.044210	0.013994	(4,5)	(4,5)	0.230403	0.123966	0.029472	(6,6)	(6,6)	0.075630	0.134916	0.019483	(5,4)	(5,4)	0.027386	0.079930	0.089910	(5,6)	(5,6)
2	0.149728	0.039928	0.014907	(4,5)	(4,5)	0.219625	0.125290	0.035601	(6,6)	(6,6)	0.072987	0.134657	0.022941	(5,4)	(5,4)	0.027263	0.081394	0.090737	(5,6)	(5,6)
3	0.152209	0.042364	0.014022	(4,5)	(4,5)	0.213680	0.129716	0.039476	(6,6)	(6,6)	0.074903	0.129974	0.020842	(5,4)	(5,4)	0.022967	0.080631	0.092752	(5,6)	(5,6)
4	0.157097	0.039950	0.013790	(4,5)	(4,5)	0.223491	0.122965	0.030205	(6,6)	(6,6)	0.072196	0.135781	0.023697	(5,4)	(5,4)	0.025731	0.080229	0.090146	(5,6)	(5,6)

Table 6

3. Test Analysis

3.1 Model Acquisition

Firstly, we will calculate the target ring's mean and standard deviation for the TR's RGB values. This is summarised in the following tables:

- Blue Ring:

	R	G	B
Mean μ:	0.025570	0.080827	0.091098
Standard Deviation σ:	0.001984	0.004306	0.002076

- Green Ring:

	R	G	B
Mean μ:	0.073333	0.131964	0.020412
Standard Deviation σ:	0.002715	0.004604	0.002196

- Yellow Ring:

	R	G	B
Mean μ:	0.220399	0.125787	0.030039
Standard Deviation σ:	0.014718	0.009016	0.003459

- Orange Ring:

	R	G	B
Mean μ:	0.154905	0.041258	0.014145
Standard Deviation σ:	0.007336	0.002149	0.000574

The mean & standard deviation were calculated using the following formulas:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(X-x)^2}{n-1}}$$ & "μ" or $x = \frac{1}{n}\sum_{i=1}^{n} X_i$

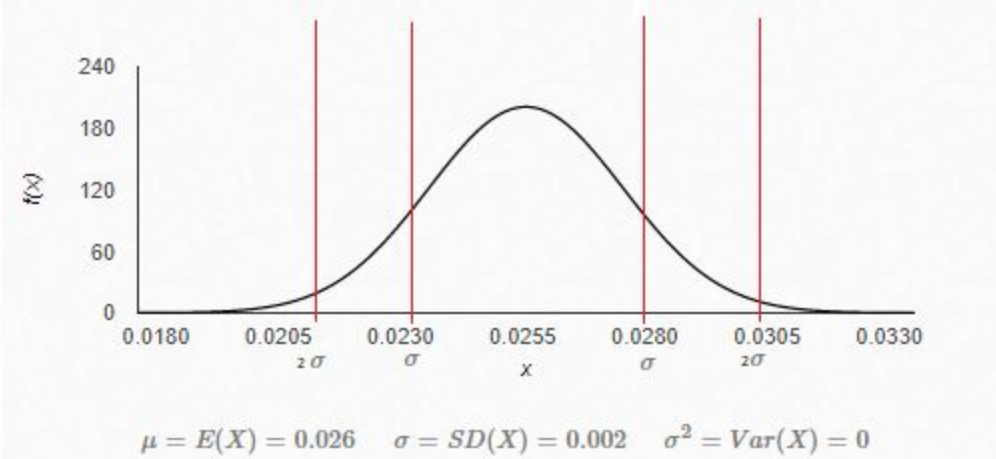
Sample Calculations for Red values of Yellow target ring:

$$\mu = \frac{1}{10}(0.218843 + 0.229607 + 0.200215 + 0.223725 + 0.229607 + 0.240310 + 0.229608 + 0.197801 + 0.23029 + 0.203984) = 0.220399$$

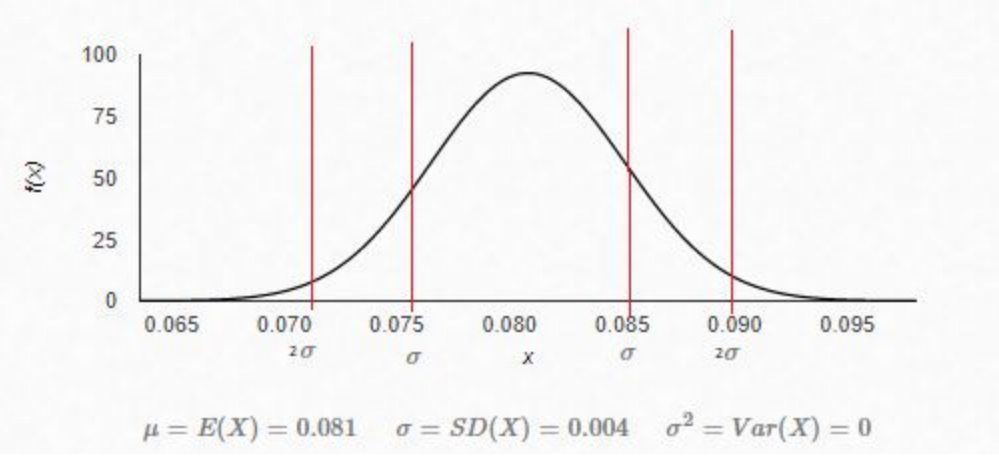
$$\sigma = \sqrt{[((0.218843 - 0.220399)^2 + (0.229607 - 0.220399)^2 + (0.200215 - 0.220399)^2 + (0.223725 - 0.220399)^2 + (0.229607 - 0.220399)^2 + (0.240310 - 0.220399)^2 + (0.229608 - 0.220399)^2 + (0.197801 - 0.220399)^2 + (0.23029 - 0.220399)^2 + (0.203984 - 0.220399)^2)\frac{1}{9}]} = 0.014718$$

We now plot the individual RGB values corresponding to each individual target rings in a gaussian plot via previously found values for their standard deviation and Mean.

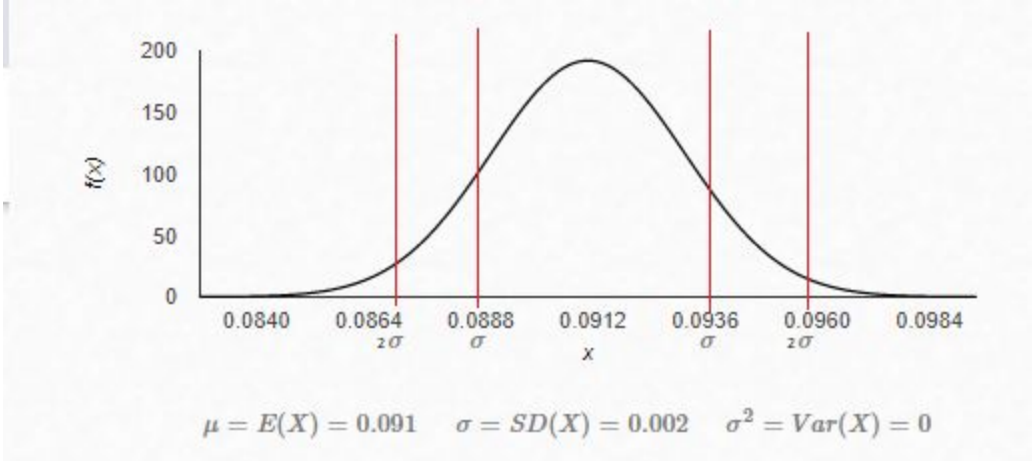
- <u>Blue Ring R value Gaussian distribution:</u>



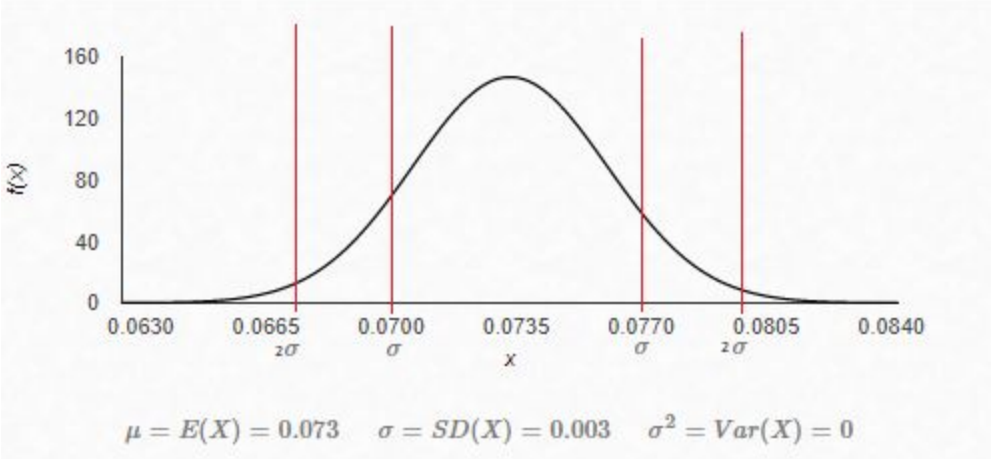
- <u>Blue Ring G value Gaussian distribution:</u>



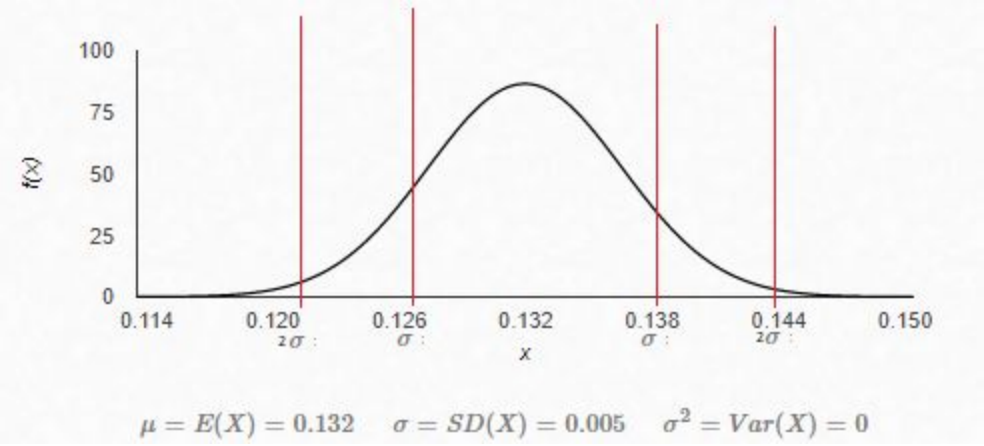
- Blue Ring B value Gaussian distribution:



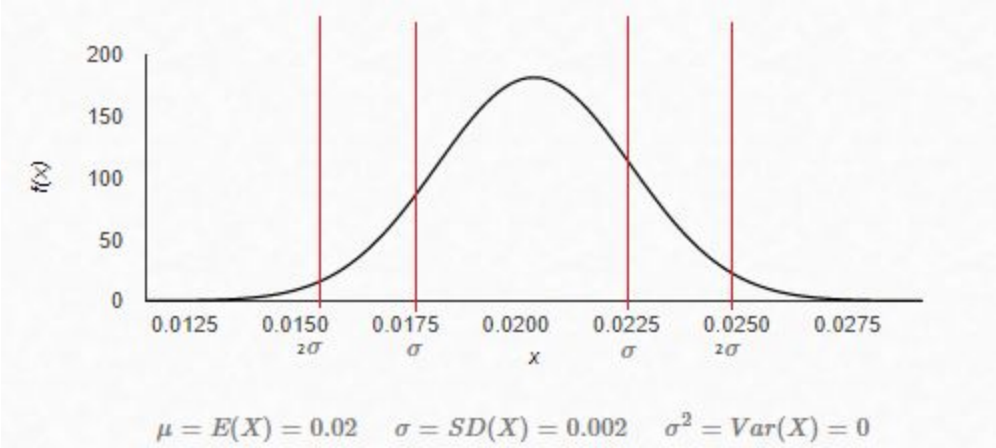
- Green Ring R value Gaussian distribution:



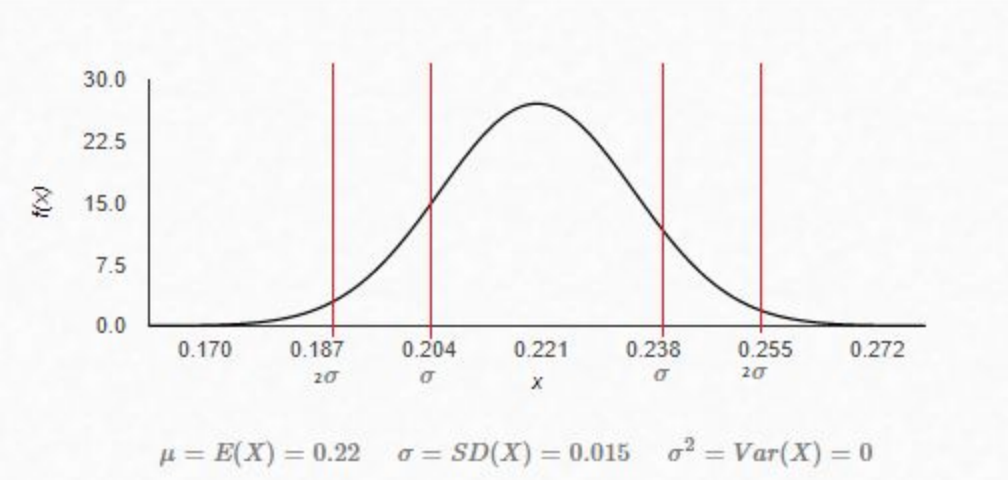
- Green Ring G value Gaussian distribution:



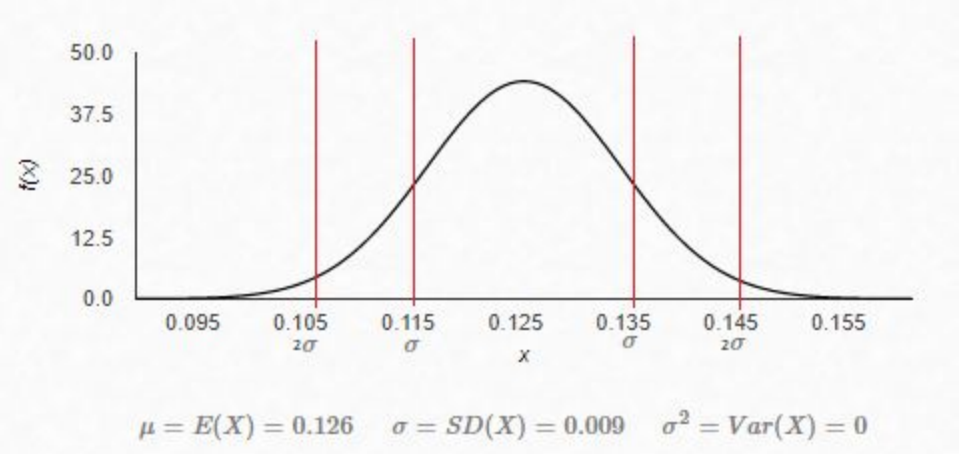
- Green Ring B value Gaussian distribution:



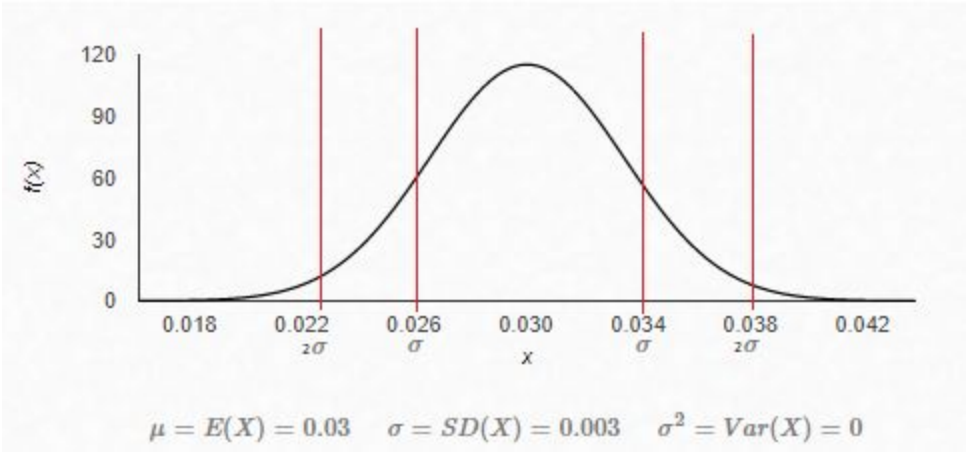
- Yellow Ring R value Gaussian distribution:



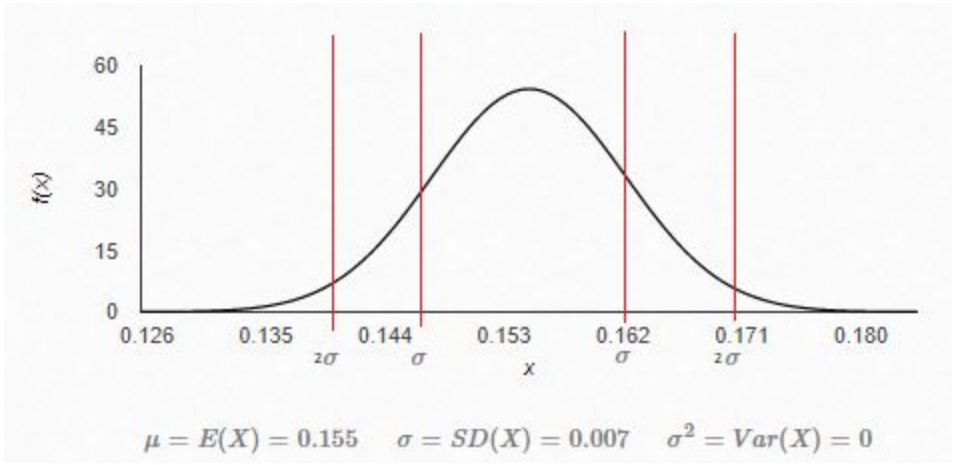
- Yellow Ring G value Gaussian distribution:



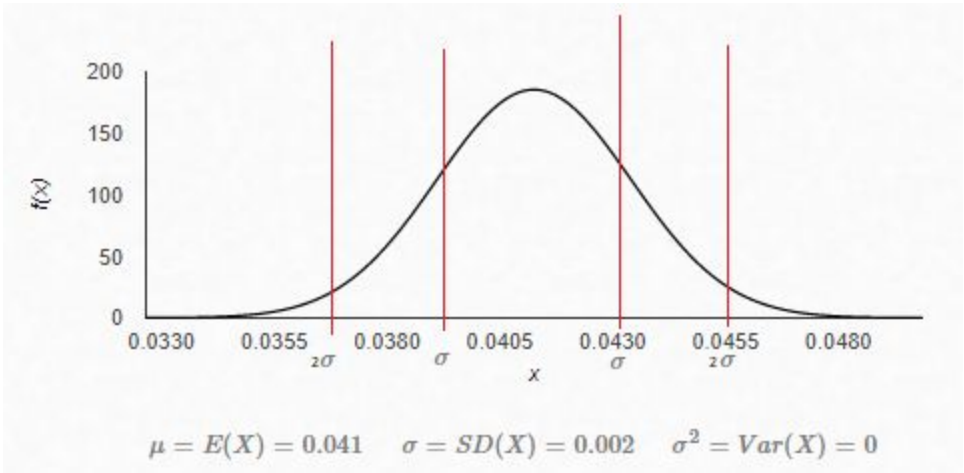
- Yellow Ring B value Gaussian distribution:



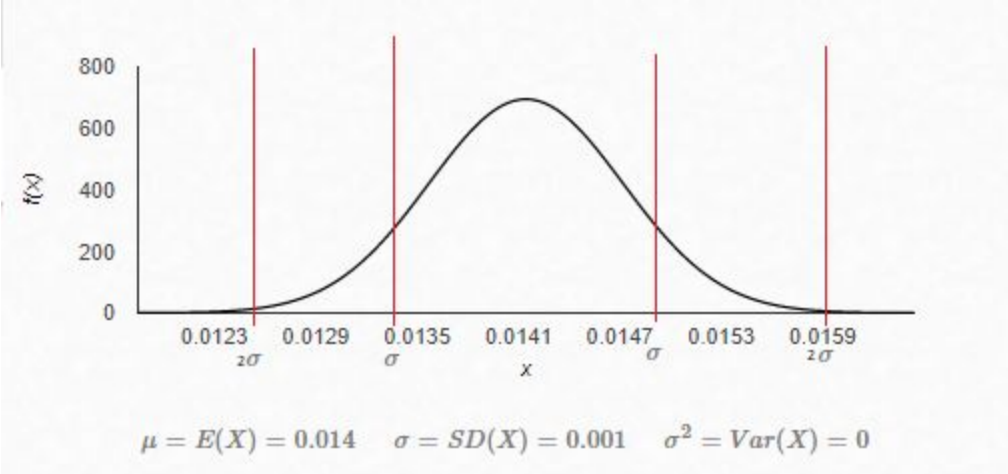
- Orange Ring R value Gaussian distribution:



- Orange Ring G value Gaussian distribution:



- Orange Ring B value Gaussian distribution:



3.2 Color and Position Identification

To determine the position of the targeted ring, we proceeded by determining the closest grid intersection of the robot when it detected the ring. To do so, we had to consider the offset of the sensor from the center of the robot and then compute the robot's current position according to the odometer values. During the trials made for color and position identification, the robot identified all the rings' color respectively and successfully determined their grid position.

Furthermore, to assess our light sensors efficiency in detecting the correctly colored rings,we calculate the euclidean color deviation ''d'' for each RGB value for each test run, and sort them in ascending order. The euclidean color deviation generally resulted in values that are approximately 0. We took a threshold of 1 standard deviation from the mean, which was sufficient to correctly identify colors. This is done via the following equation:

$$d = \sqrt{[\ (\$_R - \mu_R)^2 + (\$_B - \mu_B)^2 + (\$_G - \mu_G)^2]}$$

The Euclidean distance between the estimated position and the real position of the rings is 0 since the robot correctly returned the exact position of the targeted ring.

4. Observations and Conclusions

4.1 Are rank-ordering Euclidean distances a sufficient means of identifying ring colors? Explain in detail why or why not.

Rank-ordering Euclidean distances are a sufficient mean of identifying rng colors as it proven its capability during the data collection and test trials. However, me must only consider this solution when using the normalized means of the samples, since the ambient lightning is subject to change and thus vary the samples fetched by the color sensor.

4.2 Is the standard deviation a useful metric for detecting false positives? In other words, if the ring color determined using the Euclidean distance metric, "d", is incorrect, can this false positive be detected by using $\mu \pm 1\sigma$ or $\mu \pm 2\sigma$ values instead?

Standard deviation is a useful metric for detecting false positives, since it is rare that a sample will fall in the range of a positive result. Using a wider range of $\mu \pm 1\sigma$ or $\mu \pm 2\sigma$ would increase the chances of detecting a false positive, since it would wider the acceptable value of "d" responsible of detecting a certain ring.

4.3 Under what conditions does the color sensor work best for correctly distinguishing colors?

Both the room light condition in the lab and the distance between the light sensor and rings will influence the precision of distinguish of colors. From our test, we need to put the rings in normal room light and place it 5mm far away from the light sensor to have the best color distinguish results.

5. Further Improvements

5.1 Depending on how you implemented your color classifier, can your results be improved by using one or more of the noise filtering methods discussed in class?

There is no doubt that our color classifier may be improved via one of the filters presented in class. The sensors would provide a welcome increase in accuracy of the sensor as we've previously seen with other filters and have historically been easy to implement. Therefore, yes our results may be improved, yet this would not be a priority issue to address in our robot design since as seen

in the data, the color detection of our robot is not necessarily the primary issue. Using the normalized RGB mu values we found appropriate levels of success however under less ideal ambient light conditions such as will be present in the final lab, the noise filters will be quasi-necessary for the proper functioning of the robot

5.2 How could you improve the accuracy of your target ring's position identification?

The locating of the target ring's position was the biggest pitfall of our current design iteration. While we were able to obtain appropriate data for the lab, the robot suffered greatly in this regard during the demo. There are certain small scale modifications which can be made such as initializing all the sensors at the beginning of the operation instead of every time one is called since this takes up more computing resources of the limited hardware than necessary as well as slowing down the process which combined would not only affect the effectiveness of the robot but also the time it takes to operate which is capped during the final project. Avoiding that needless computing overhead via software changes can also be done via limiting the amount of threads our code utilizes. As we've seen this not only drastically affects the performance of the processor of the robot, but also affects the stability and the actual calling to the threads which are required for the proper identification of the location for the rings. There is also the possibility of combining certain classes, particularly some of the ones involving the sensors which would also remove much of the overhead within the software. Improving these small details and optimizing the software has large resonating effects on the success of the robot. The largest issue with the robot was its inability to effectively navigate the search area due to the odometer, at its core, being dysfunctional. There are numerous points during the software where there is the possibility that an integration conflict vastly falsifies the odometer and that can be reevaluated such as the localizing and the correction. The method in which the localizing interacts with the odometer can be modified to ensure its reading of values and adjusting does not throw off the odometer without the possibility of future correction fixing the issue. The correction itself can be improved as well, simple hardware changes such as extending out the two sensors to detect the lines would provide a more accurate reading by which the robot may adjust along with software optimizations within it such as reducing any hard coding.