TEAM 11

MCGILL ENGINEERING DESIGN

ECSE 211

ECSE 211 LABS

LAB 5- SEARCH AND LOCALIZATION

03/02/2018

ECSE 211 TEAM 11

LUKA JURISIC - 260732284

BRYAN JAY - 260738764

VOLEN MIHAYLOV-260746982

ENAN ASHADUZZAMAN-260805923

TIANYI ZOU-260724913

PATRICK GHAZAL- 260672468

ROBOT WITH SEARCH AND LOCALIZATION– OVERVIEW AND ANALYSIS

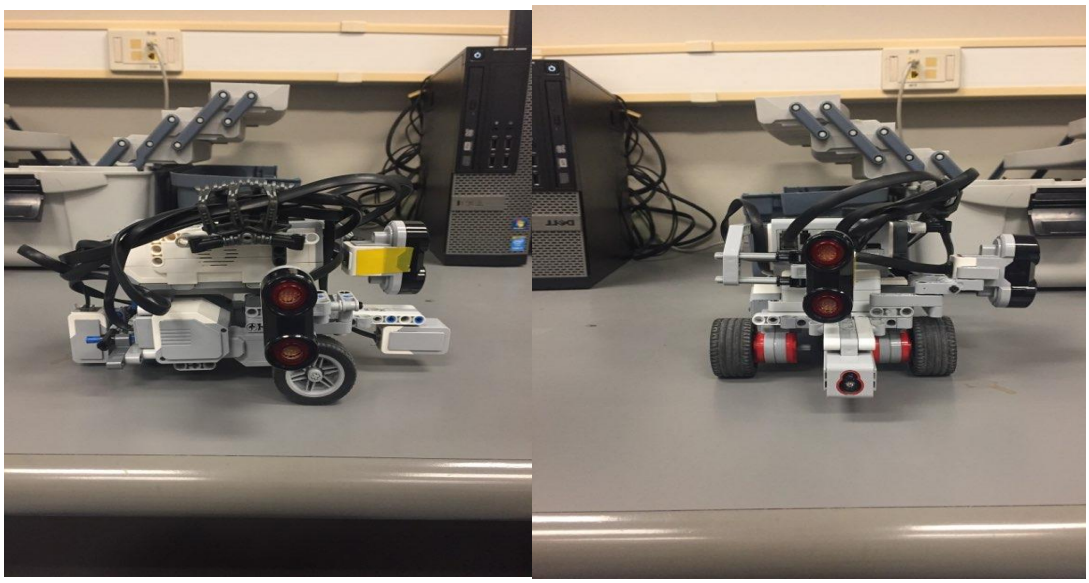
Design Evaluation

Our robot is equipped with 2 large EV3 motors. In deciding between the EV3 and NXT motors, we evaluated that the precision and accuracy provided by the EV3 motor would be much more appreciated for a lab that required a high precision for measurements. There were 2 major hardware decisions made during this lab.

i. This lab required the unifying of 3 previously separate lab teams, and thus each team had brought with them their own robots. The team had to decide which architectural design was most useful to the requirement of this lab.
ii. The 2nd integral design issue regarded the positioning of the ultrasonic and light sensors.

Concerning the 1st decision, it was determined that each robot had too large a wheelbase, which would serve as a hindrance in this lab due to the added searching requirement. Thus, a new robot with a smaller wheelbase was designed completely.

Regarding the 2nd, the robot consists of two ultrasonic sensors placed 90 degrees apart, and two light sensors, one serving to detect the colour of an object and the other used to localize the robot. For localization, one light sensor was positioned at the back of our robot as this would make it easier to detect black lines when rotating in place during localization. The 2nd light sensor was placed in front, slightly protruding ahead of the ultrasonic sensor, as it would have to get very close to the object to detect its colour. A discussion was held as to whether placing the light sensor on top using a supporting beam was a viable option and it was concluded that although it was indeed applicable, the added hardware endeavor was deemed unnecessary as the current frontward position was also effective. The robot's final hardware configuration is illustrated in figures 1 and 2 below



Figures 1 and 2- Robot Design (Side and Front Views)

Workflow

The initial steps regarding this particular lab was the complete redesign of the current hardware architecture. Mr Enan is the hardware team leader within our group and he handled this process. Mr Tianyi, as a hardware engineer, assisted. This took approximately 45 minutes. Following this, the software team, consisting of Mr Jay, Mr Ghazal and Mr Volen, proceeded to outline the pseudocode for the searching path. Mr Jurisic assisted. The team then proceeded to implement this pseudocode with Mr Ghazal taking charge as software lead. Once all members were satisfied with the robot's performance, sufficient data was then collected for the lab report. This software implementation took approximately 12 hours. Following the completion of the software and hardware components, Mr Jurisic, as documentation head, took the lead in the writing and formatting of the lab report. This took approximately up to 4 hours.

Software

It must be noted that the software for this lab proved to be more challenging than for the previous ones. Indeed, although a portion of the code was already written, each code-writer has their own habits and procedures, and this difference is exacerbated when code from different sources is brought together, as was done here. One example of this dichotomy is the Odometer class. The one we had originally proved to be occasionally faulty and was therefore

replaced by the class written by another sub-team of the group. However, the data being transmitted to and from the Odometer class in each case was not handled the same way (e.g. radians/degrees, unit lengths/real lengths) and this caused delay as we had to fix issues such as this one.

Furthermore, in this lab, the successful implementation of the software involved solving many sub problems and subsequently synthesizing the solutions to perform the correct task. To elaborate, there were 4 main sub problems that required consideration, they are listed below.

1. Localization within the grid using both ultrasonic and light sensors
2. After localizing, navigation to the lower left of the search area and hence with applying the search algorithm within a 5-minute time restriction.
3. Ability to detect 2-5 blocks and sample their colour.
4. Moving to the upper right corner of the search corner to satisfy the requirements.

The first problem was very simple to tackle as it was sufficient to utilize the existing code from the previous localization code. A small change was made in which the robot would be able to choose between rising or falling edge localization based on the reading of the front sensor. If it reported a large initial distance, falling edge would be chosen, and vice versa.

The difficulty of this lab certainly manifested itself in the tackling of the 2 and 3rd subproblems; figuring out an efficient and effective traversal method to carry out a search within 5 minutes. 3 methods of traversal were actively discussed before a final method was chosen. They are detailed below.

1) Spiral Traversal

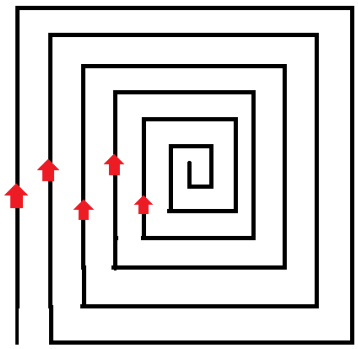


Figure 3: Spiral traversal illustration

The robot would start in the lower left-hand corner and traverse its way through the search grid in a spiral fashion, avoiding blocks as depicted in the diagram. Every lap the robot would shorten its search area by 10 centimeters (to avoid missing any cubes between the spirals) and continue scanning in front of it until it would either find the targeted cube or reach the middle. However, this method is inefficient as the required travelling distance is simply too great; it would take greater than the allotted 5-minute time restriction. Therefore, we had to discard this idea.

2) Snake Traversal

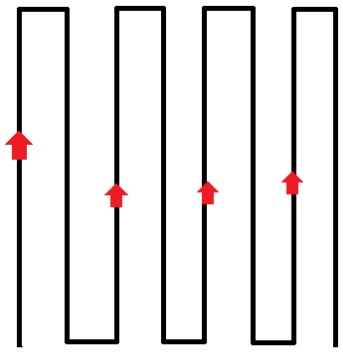


Figure 4: Snake Traversal illustration

The second search method considered is the "Snake Search Method"- involving going through the grid in a zig-zagging 90-degree pattern (figure) with each path being separated by a 10cm distance. This method is shorter than the spiral method, however complications would arise in certain scenarios; issues arise when the cubes are next to each other, and multiple possible cube positions when it comes to the path complicate the pathing algorithm by a huge amount. Therefore, this idea was discarded.

3) Zigzagging Method

A third search method was as well proposed, consisting of going through the field in a full zig-zagging pattern (Figure 5). The robot would detect the cubes in front of it just like in the previous two methods and would round the detected cubes from their left using a secondary ultrasonic sensor. However, it proved in theory to be even more difficult to code, seeing as the path is not constituted of 90° simply angles it is harder to verify at which point on the path to robot is, since its pathing will be often broken when rounding the cubes that it will have detected in from of it. Furthermore, the same issue arises as with the last method, when there is a cube behind another.

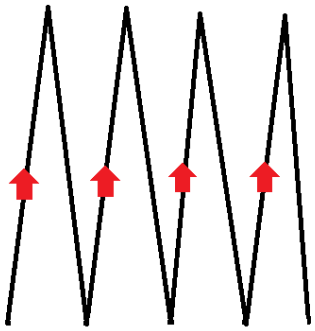


Figure 5: Zigzag traversal illustration

These considerations finally led us to the idea of simply traversing the periphery of the search area, and only enter it when a block is detected by the right mounted US sensor. This solution involved very minimal turning and removed the added complication of having to consider obstacle avoidance. The method revolves around allowing the robot to drive along the edge of the grid, using the front sensor to navigate, and have the right sensor serve as a detection for when to turn into the grid. Upon seeing a block and turning inwards, the robot would drive until the distance to the object was 3.0cm.

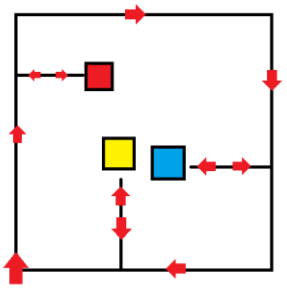


Figure 6: Peripheral traversal illustration

The light sensor would then be able to detect the colour of the object. Figures 7,8,9,10 and 11 below summarize the functionality of our software design.

Figure 7: Navigate travelTo functionality

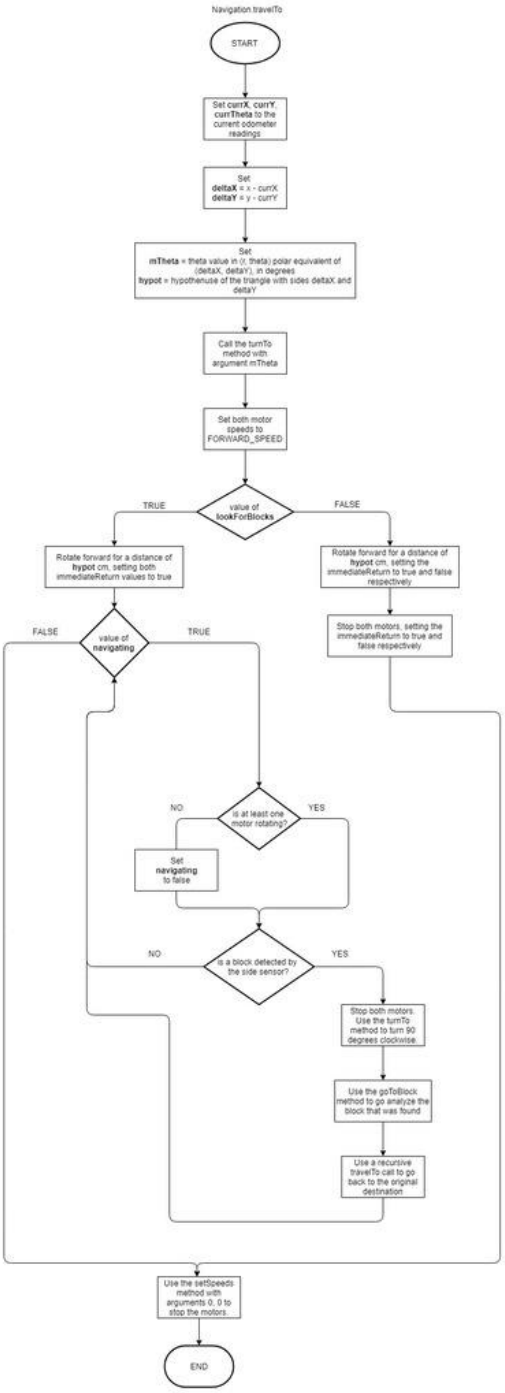


Figure 8: LightLocalization functionality

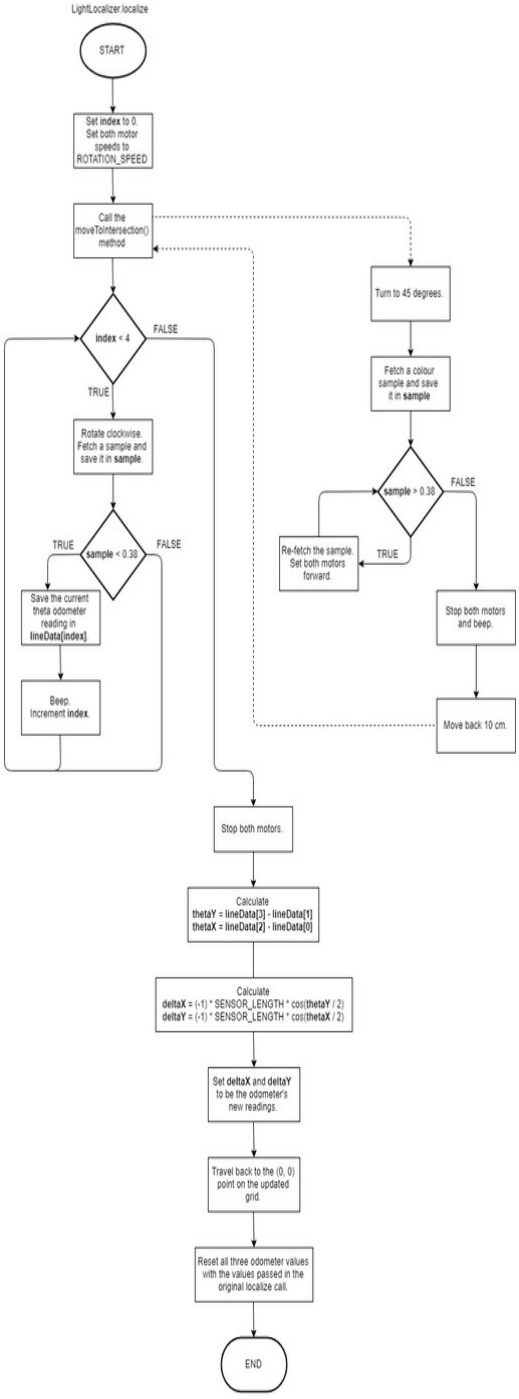


Figure 9: USLocalizer Functionality

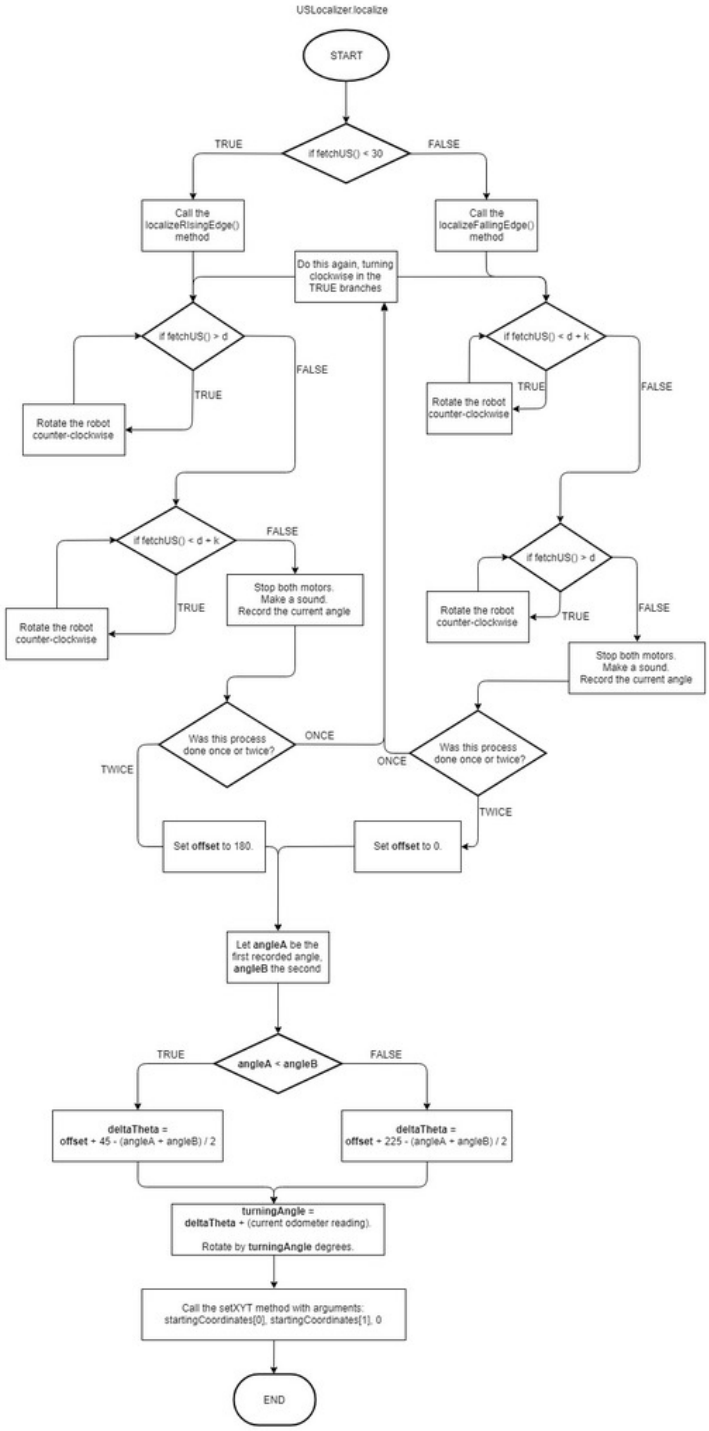


Figure 10: Lab5.main Functionality

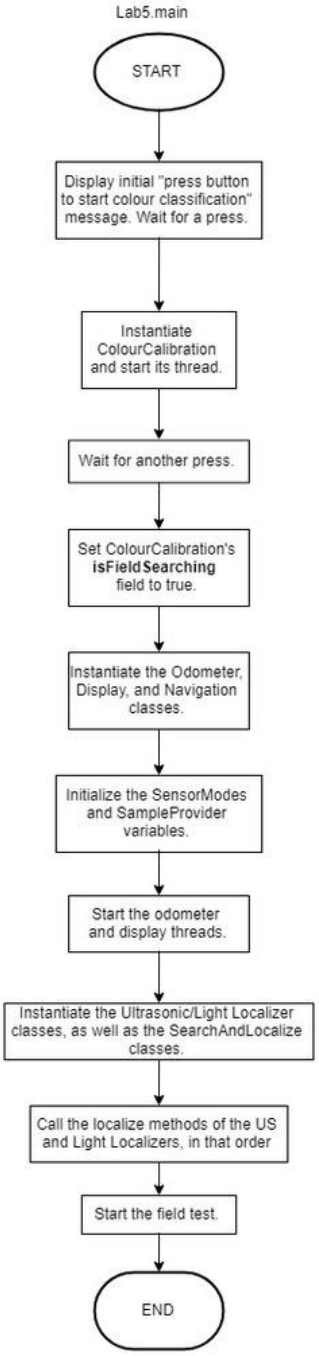
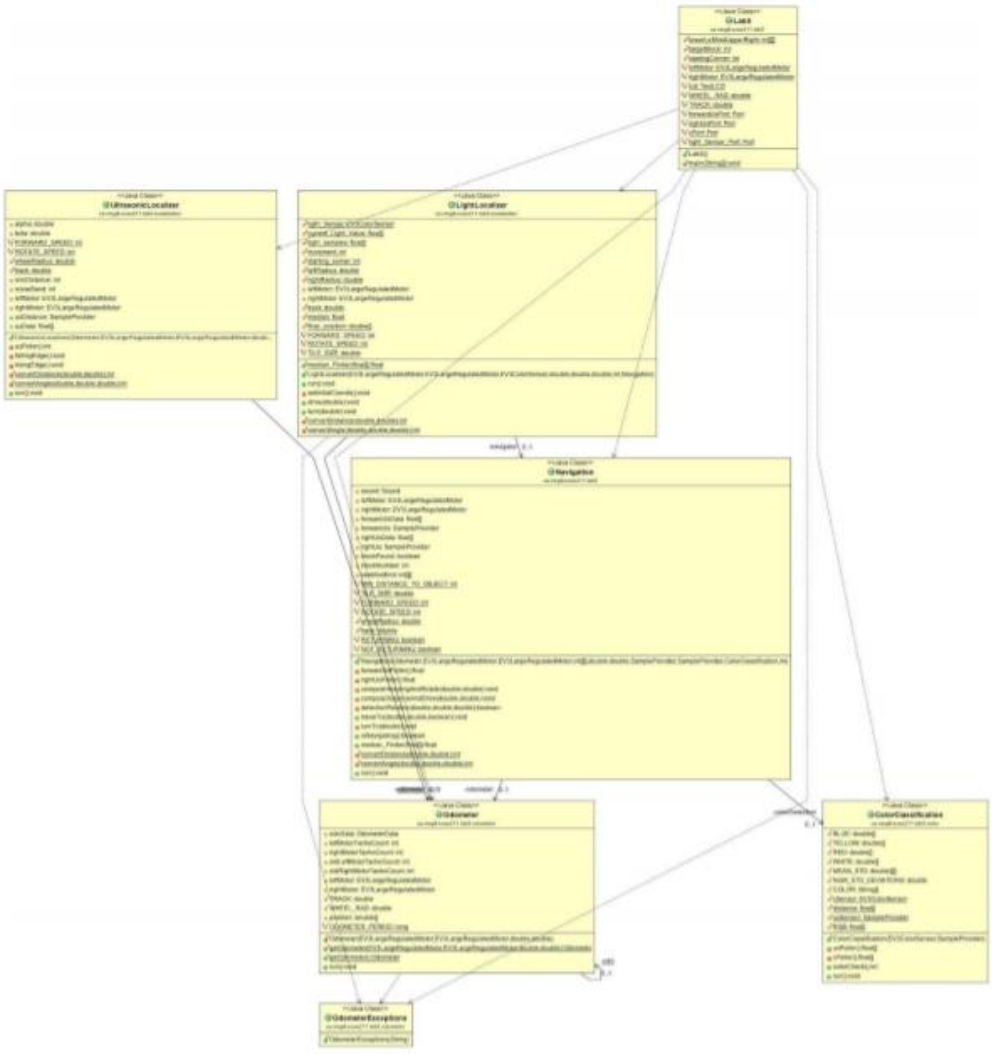


Figure 11: Class Diagram dependency illustration



Test Data & Test Analysis

Table of Euclidean distance of all recorded RGB values in ascending order

Euclidean distance (cm) Trial	Red	Blue	Yellow	White
1	0.0043	0.0029	0.0150	0.0
2	0.0047	0.0038	0.0191	0.0105
3	0.0104	0.0047	0.0248	0.0761
4	0.0105	0.0072	0.0271	0.0370
5	0.0124	0.0075	0.0365	0.0092
6	0.0134	0.0091	0.0391	0.0014
7	0.0353	0.0075	0.0	0.0332
8	0.0407	0.0120	0.0816	0.0056
9	0.0458	0.0170	0.0819	0.0420
10	0.0459	0.0173	0.1727	0.0401

Table of Euclidean distance between the estimated position (TPEx,TPEy) and real position (TPRx,TPRy)

Trial	TPEx	TPEy	TPRx	TPRy	Euclidean distance (cm)
1	5	5	5	5	0
2	4	6	4	6	0
3	6	5	6	5	0
4	4	5	4	5	0

Note For the (TPEx, TPEy) values, we chose the closest tile coordinate based on the odometer. For example, if the odometer got the value of (58.5,56.4), we would take (2,2) as the value of (TPEx, TPEy).*

Model Acquisition Test

Through trial and error, the working range of the sensor was determined to be approximately between 0.50 and 2.50 cm.

	RED			BLUE		
	R	G	B	R	G	B
1	0.13745	0.00890	0.01060	0.01330	0.04282	0.05676
2	0.14905	0.01450	0.01170	0.01833	0.03333	0.04345
3	0.17575	0.01160	0.00990	0.01676	0.03925	0.04234
4	0.12765	0.02050	0.00760	0.02450	0.04200	0.05676
5	0.10295	0.00890	0.01864	0.01676	0.03925	0.04245
6	0.18245	0.01960	0.01088	0.01960	0.04366	0.05676
7	0.17575	0.01250	0.01066	0.01833	0.04200	0.04345
8	0.13745	0.01160	0.01066	0.02250	0.03876	0.04245
9	0.10295	0.01360	0.01146	0.01922	0.04506	0.05166
10	0.13333	0.01250	0.00990	0.01960	0.04200	0.04780
Mean	0.142478	0.01342	0.0112	0.01889	0.040813	0.048388
Standard Deviation	0.02854635	0.003925642	0.00284911	0.003101441	0.00332488	0.00647378

	Yellow			White		
	R	G	B	R	G	B
1	0.26547	0.12657	0.02456	0.23450	0.17355	0.12078
2	0.21186	0.15394	0.01978	0.21465	0.16545	0.13456
3	0.20666	0.11780	0.02456	0.20456	0.18678	0.12456
4	0.24500	0.10787	0.02057	0.21245	0.15678	0.10678
5	0.23450	0.11373	0.01978	0.23765	0.16765	0.08976
6	0.17890	0.10567	0.02040	0.20897	0.18765	0.09086
7	0.26547	0.14920	0.01890	0.19856	0.19024	0.10456
8	0.20666	0.11373	0.01568	0.18745	0.16588	0.12157
9	0.19340	0.09870	0.02658	0.19856	0.13476	0.13933
10	0.18900	0.10567	0.01890	0.21456	0.18452	0.14523
Mean	0.219692	0.119288	0.020971	0.211191	0.171326	0.1177997
Standard Deviation	0.0311792	0.018671575	0.00328497	0.015634401	0.01721893	0.01936288

Observations and Conclusions

Are rank-ordering Euclidean distances a sufficient means of identifying block-colors a sufficient means of identifying block-colors?

The use of a rank-ordering Euclidean distance system to detect colors is advantages in the sense that it provides a single measure of closeness when using 3 variables (R, G, B intensities). However, there is one main imposition that questions that practical implementation of this system.

This system is inhibiting itself as it does not allow us to how close each of the 3 of R, G and B intensities are close to the tested colors individually. This could give rise to problems in situations where 2 colors whose R, G and B values are very similar. In scenarios where many colors need to be sampled, this Euclidean distance rank ordering system will certainly encounter errors.

Consequently, rank-ordering Euclidean distances as a color detection means is subject to drawbacks and its practical limitations of implementation subsequently indicate that it is not a sufficient means of block color detection.

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

Standard deviation is a useful metric on its own as it factors in the variability of data associated with the mean; it indicates how spread out the data collected is. Therefore, using a mean and standard deviation metric would allow us to see whether an incorrect Euclidian distance sample obtained was tolerable by considering the spread of the data. For example, assuming an incorrect sample, the metric tells us that 68% of the data is within 1 standard deviation of each side of the mean and 95% of the data is within 2 standard deviations. Therefore, if there was a false positive within the data, this metric would allow us to evaluate how tolerable this Euclidean distance is given the spread of data. Because of this, standard deviation, coupled with the mean, proves a useful metric at detecting and abating false positives from the testing data.

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

Initially, the positioning of our light sensor proved a hindrance in allowing correct light sensor readings. It was positioned too inwards into the NXT brick and this meant that it was not able to get close enough to a block to accurately measure its colour. Further testing indicated that the light sensor worked optimally when within 1 centimeter of the object. This is because at large distances away from the block allowed for ambient light intrusion, disturbing the values returned by the sensor. On the other hand, distances less than 1 cm were ineffective because the sensor's floodlight was not able to encompass a large enough surface area of the object to consistently measure accurate values.

Further Improvements

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?

The color classifier on our design was mounted in the front of the robot, and the light sensor detected colors as it approached the object. t. Our light sensor was always the same distance away from the top of the blocks which kept the scanning conditions very constant. As a result, we did not have much use for a filtering method since the only "noise" in the measurements came from the ambient light condition.

How could you improve the accuracy of your target blocks position identification?

The accuracy of our current position identification could be improved by implementing two forward facing US sensors to determine the location of a block and approach it precisely. Incorrect approach by a slight angle deviation could be corrected and improve the chance of not missing a block.