Beta Test Analysis

Project: LIBERTY

Task: Review Results from the Beta Demo

Document Version Number: 1.0
Date: 20/11/2017
Author: Bill Zhang
Editor: Andi-Camille Bakti
Edit History: https://github.com/Gabetn/DPM_01_Project_Documentation

McGill

TABLE OF CONTENTS

1.0 Observations

During the beta demonstration, the robot was placed on the corner 1 whose coordinates were (7, 1) as starting points with a battery level of 8.0V. After the data was passed by the WiFi component, the robot appeared to perform ultrasonic and light localization successfully. Upon the completion of localization, the robot turned to the true zero (defined as the direction from 0,0 to 0,1) without a minimal angle strategy. Up to this point, everything had worked as expected. However, when the navigation was initiated, the robot drove turned towards the wrong direction and thus resulted all sequential and concurrent steps failed.

2.0 Potential Components Flaws

After the observing the behaviors of the robot and repeating the test using the same dataset, some classes are targeted for potential flaws. Their names and the rationale behind it are as following, ordered by most probable suspect:

1. Navigation: The robot performs successfully until the navigation is initiated, therefore, there is a great probability that navigation is the prime suspect and has faults in its design.
2. Ultrasonic Localizer: The robot appears to performs this step well. However, it could potentially fail to set the correct odometry data hence affecting Navigation.
3. Light Localizer: As for the Ultrasonic Localization, the same reason applies here.
4. Odometry Correction: Both localizers involves detecting line and the movement of the robot at the starting point would inevitably crosses a few lines, therefore, it could affect the odometry correction to set incorrect data in the odometer.
5. WIFI jar file: The wifi component could pass data in a different or invalid format that is used in the program. It could potentially cause that the robot calculates wrong coordinates.

3.0 Test Trials Table

To find the reason of failure, two kinds of tests are conducted. First, unit test for each potential flaw class, surprisingly, all classes passes the unit tests. Then, a series of integrating test is run with different combinations of classes. The content and results are shown following.

Name	Navigation	Ultrasonic Localization	Light Localization	Wifi	Odometry Correction	Result
Trial 1	Y	Y	Y	Y	Y	Fail
Trial 2	Y	Y	Y	N	Y	Fail
Trail 3	Y	Y	N	N	Y	Fail
Trail 4	Y	N	N	N	Y	Fail
Trail 5	Y	N	N	N	N	Success
Trail 6	Y	Y	Y	Y	N	Success

Table 1: Beta Test Results using the same data set as provided.

As the test results show, as soon as the odometry correction is not in use. The test is passed successfully. Therefore, the odometry correction is the class that directly caused the failure. A more detailed analysis is shown in the next section.

As the rest results suggest, the error is in the odometry correction class, however, this class passes the unit test. Therefore, the pre-conditions to trigger the odometry correction is analyzed. It is found that the robot turned to true zero after the localizations as designed. However, it did not turn the minimal angle to achieve it. Additionally, the turn method in the program creates decent amount of errors when turning a large angle. Thus, the turn is not accurate which triggers odometry correction. Because the robot will inevitably crosses a few black lines along the process, the odometry correction is detecting more lines than it is supposed to do which set the incorrect data in the odometer. Therefore, when the navigation is initiated, the odometer is flawed. As a result, the robot drove off course.

4.0 Proposed Solution

There are some parts of the program that we can improve regarding this failure:

1. First, the physical dimensions of the robot can be re-measured to update the constants in the program to turn and drive. There are some changes made in the hardware design after the initial measurements, so the constant might need updating.
2. Second, a minimal angle algorithm can be implemented in the navigation class to reduce the angle required to turn which will decrease the accumulative errors.
3. The odometry correction does not have to be started in the very beginning because of the possibility of accumulative errors. Therefore, the odometry correction can started later when the robot has started moving towards the first coordinates to completely eliminate the possibility of setting the incorrect odometer data.