Steven Cangul (260744412) and Noah Levine (260684940)

# Lab 2: Odometry

## Data

Table 1: Odometry Without Correction

| Odometer X (cm +/- 0.05cm) | Odometer Y (cm +/- 0.05cm) | Actual X (cm +/- 0.05cm) | Actual Y (cm +/- 0.05cm) | Error X (cm +/- 0.05cm) | Error Y (cm +/- 0.05cm) |
|---|---|---|---|---|---|
| -15.5 | -15 | -15 | -14.7 | 0.5 | 0.3 |
| -13.5 | -17 | -14 | -15.3 | 0.5 | 1.7 |
| -12 | -14 | -15 | -13 | 3 | 1 |
| -12.5 | -15 | -14.5 | -14 | 2 | 1 |
| -15.7 | -14 | -15.3 | -14.2 | 0.4 | 0.2 |
| -16.5 | -13.8 | -16.2 | -14 | 0.3 | 0.2 |
| -12.5 | -14 | -14.5 | -13.4 | 2 | 0.6 |
| -13 | -14.7 | -14 | -13.5 | 1 | 1.2 |
| -13.8 | -14.2 | -14.2 | -13.9 | 0.4 | 0.3 |
| -14 | -15.3 | -14.1 | -15.5 | 0.1 | 0.2 |
| | | | **MEAN (cm)** | **1.02** | **0.67** |
| | | | **STANDARD DEVIATION (cm)** | **0.92** | **0.5** |

Table 2: Odometry With Correction

| Odometer X (cm +/- 0.05cm) | Odometer Y (cm +/- 0.05cm) | Actual X (cm +/- 0.05cm) | Actual Y (cm +/- 0.05cm) | Error X (cm +/- 0.05cm) | Error Y (cm +/- 0.05cm) |
|---|---|---|---|---|---|
| -14.7 | -14 | -15 | -14.2 | 0.3 | 0.2 |
| -15.3 | -14.1 | -15.1 | -13.7 | 0.2 | 0.4 |
| -14.9 | -14.7 | -15.3 | -15 | 0.4 | 0.3 |
| -15 | -15.4 | -15.5 | -15.6 | 0.5 | 0.2 |
| -14 | -13.9 | -14.2 | -14.4 | 0.2 | 0.5 |
| -15.1 | -14.5 | -15.5 | -14.9 | 0.4 | 0.4 |
| -16.3 | -12.5 | -14.8 | -14.5 | 1.5 | 2 |
| -13.7 | -14.5 | -14.2 | -14.3 | 0.5 | 0.2 |
| -14.3 | -14.7 | -13.7 | -14.5 | 0.6 | 0.2 |
| -14.3 | -15.2 | -14.3 | -14.7 | 0 | 0.5 |
| | | | **MEAN (cm)** | **0.46** | **0.49** |
| | | | **STANDARD DEVIATION (cm)** | **0.39** | **0.52** |

## Data Analysis

PART A

The standard deviations for the x and y measurements without correction were 0.92cm and 0.5cm respectively. When correction was implemented, the standard deviations computed for x and y were 0.39cm and 0.52cm respectively. The standard deviation for x decreased significantly (by 0.53cm) and very slightly increased for the y values (by 0.02cm). Theoretically, the standard deviation should decrease in both positions when correction is added. This is because the standard deviation measures the dispersion of a given set of data. That is, a large standard deviation means the data results are dissimilar to each other while a smaller standard deviation means the data is more similar to one another. When we apply correction to the odometer, we expect more accurate results and thus all errors should be pretty small and similar to each other. This is what caused the decrease in deviation for the x values. The increase in y can be attributed to many factors. Firstly, it is important to note that the increase was very small. Secondly, the y values had been obtaining more accurate results than the x values without correction being applied. Thus, there was less room for improvement in the y values as they had already been quite accurate. Finally, if we were to analyze the data set, we can recognize one obvious outlier, trial 7. Whereas all of the other trials had measured y error values within the 0.2-0.5cm range, trial 7 measured an error of 2cm. The reason for this result could have been due to a slip after the last correction was applied, to a malfunction of the light sensor or perhaps another reason. Whatever the case, this trial skewed the results and our calculated standard deviation. If we were to recalculate the standard deviation with trial 7 removed, we would obtain 0.21cm, a decrease of 0.29cm.

PART B

With the implementation of a correction method to update the variables making up the odometer, we expect the error in both the x and the y position to be smaller. In fact, light correction allows us to know where the robot **actually** is with respect to the starting line. In fact, we know where all the lines are situated and this is a very important piece of information when trying to perfect such a system. Knowing where the robot actually is and where it thinks it is based on its integrated odometer, we could use these results to calculate a correction factor and basically tell the odometer where the robot is physically rather than letting it think that it is located somewhere else. This correction would decrease the chances of error based on various factors such as the wheels moving without the frame of the robot moving (i.e. wheel slip).

## Observations and Conclusion

The observed odometry error without a correction being implemented is not tolerable for larger distances. We would expect error to grow linearly. That is if the robot were to have travelled a square of dimensions five times larger than what was requested for this laboratory, we would expect the errors in both the x and y positions to have increased by a factor of five. Thus, the expected mean difference in x and y would be 5.1cm and 3.35cm respectively. We expect this linear relationship because the robot is travelling at a constant linear velocity. Thus, we expect the amount of slippage (factor that causes error) to be proportionally related to the distance travelled. The larger distance it travels, the more it slips. While the mean errors might not be that far off for short distances, they can be very large for large distances. This is why correction is implemented. The use of correction eliminates this problem as we would expect the mean errors in x and y to remain the same if we were to increase the distance.

## Further Improvements

Part 1

There are many was of creating a robot that is able to efficiently and accurately track its own position in space. A vast selection of sensors such as GPS and dedicated systems allow this to be realizable without much effort. Our robot, however, used the rotation of its wheels as its sensor. In fact, using simple physics, one can write code to detect the angle by which the motors rotate and calculate its arc length (i.e. Distance travelled). However, slip is one of the components that must be taken into consideration when opting to use such a method. In fact, if slip is indeed present when using the robot, the wheels are indeed turning but the robot itself isn't moving physically. This would confuse the robot's odometry system, thinking it is somewhere else when in fact it is not.

One simple way to solve this issue is to adjust the code the robot is running on to slowly and gradually increase the speed of the wheel. Rather than forcing the wheels to pick up speed quickly and possibly losing grip due to the physical properties on the materials in contact or the conditions of the surfaces, making the wheels accelerate smoothly would greatly decrease the chances of error. This is in fact relatively easy to implement using the **smoothAcceleration (boolean yes)** method available in the EV3 library.

Another way of going around this issue is to implement what is called the cross-coupled controller. Essentially, this controller continuously verifies the tachometer readings sent in by each motor and compares them to each other. If one of them is going too fast and has to potential to slip as it is held back by the other wheel, it is slowed down proportionally to its speed. The opposite happens if one of the wheels is seen as too slow.

Part 2

I. One of the improvements we could have made to our robot if we had two light sensors at hand would be to accurately correct the angle of the path which the robot thinks it is travelling with. If the robot had two light sensors, both placed in a V configuration in front of the robot, we could use mathematics and physics to accurately calculate the angle through which the robot has changed its direction. If both sensors are able to detect the black strip at exactly the same time, then we know that the robot is facing the black stiped line either perpendicularly or exactly at an angle of 45 degrees, on the diagonal of a square. Logic statements in the coding could determine which case it falls into based on how much the wheels rotated and in which direction they rotated. If one of the sensors gets triggered before another however, this would suggest that the robot is in fact not travelling in a straight line or diagonal to a square. Calculating the time between each trigger and the speed at which the robot is moving, we could use physics to calculate the angle in which it is travelling. The sensors also check whether it crosses the line running parallel to the direction of motion to avoid crossing. A more detailed explanation of this is available at https://app.assembla.com/wiki/show/DinaBOT/Odometry_And_Localization.

II.

If our robot has only one sensor mounted to its frame, it renders the angular correction to be less efficient. However, this does not mean that it is impossible to implement correction using a single light sensor. One of the methods that could be used is to verify the behaviour of the robot as it crosses two consecutive lines. Knowing the speed of the robot and the time between each trigger, we can calculate the angle described by the path of the robot while crossing from one line to the other and apply a proportional correction based on the results obtained and the desired trajectory direction.