Perimeter Calculation using Bump Sensors and Rotarty Encoders of 3Pi+ Robot

2180087 2212125

***Abstract*—The aim of this study was to determine how viable the bump sensors and rotary encoders of Pololu 3Pi+ robot are for distance measurement. Through an intensive investigation of bump sensors and rotary encoders we have explored how to drive the robot in a straight line and reduce the error in the rotary encoders. We evaluated its performance by analysing the perimeter of a box with the use of bump sensors and then by removing the plastic cover of the robot and measuring the same data with the help of IR sensors. We found our system performs with the least amount of error when we make use of the IR sensors for collision detection.**

I. Introduction

Robots are ubiquitous. They are used in almost all the industries such as health, agriculture, automobile, domestic usage and many others. The precision and the speed of the robots are dependent on the applications of the robot. Commercial robots need to have the least amount of failure while also being commercially viable for large-scale production, this requires them to be cheap to produce. One way to reduce this cost is by having the least amount of sensors that can perform the task at hand adequately. This paper aims to investigate the error accumulated by the change in encoder count when a bump is detected using the bump sensors of the Pololu 3 Pi+, understand the source of the error, and find ways to minimise the error when the 3Pi+ robot is used in a particular task.

Nowadays, hands-free vacuum and mop robots such as the Roomba have seen widespread success having sold around 40 million units over the past twenty years worldwide. The Roomba cleaning robot uses bump sensors to detect collisions such as chairs, tables etc., in the house and wheel encoders help the robot to take a turn and move to the next area to clean [1]. Collision detection and taking turns based on the detection are the two fundamental tasks and the performance of these fundamental tasks is vital to vacuum and mopping the floor of a house. In our experiment, the objective is to analyse how much the bump sensor has an effect on the rotary encoders by measuring the error accumulated in encoders while the robot measures the perimeter of a square box, a test environment.

We use 3Pi+ Polulu, a high-performance mobile robot with an Arduino-based ATmega32U4 MCU microcontroller [2] in our experiment as this robot is useful for beginners who want to learn and explore robotic systems/electronics. It has rotary encoders to calculate the rotation of the wheels, line sensors, bump sensors, a compass and a full Inertial Measurement Unit (IMU), but we use only bump sensors and rotary encoders in our experiment.

In this study, we operate our robot to navigate along the edges of the square box to measure the perimeter of the box. Whenever a robot detects a collision, the robot will take a turn and travel along the edge of the test area. The distance travelled by the robot is calculated based on the encoder counts of the wheel. The speed of the robot will cause the error in the encoder counts due to collision and we explore the ways in which we can reduce the error in encoder counts and what are the other sources of error while measuring the perimeter of the test area.

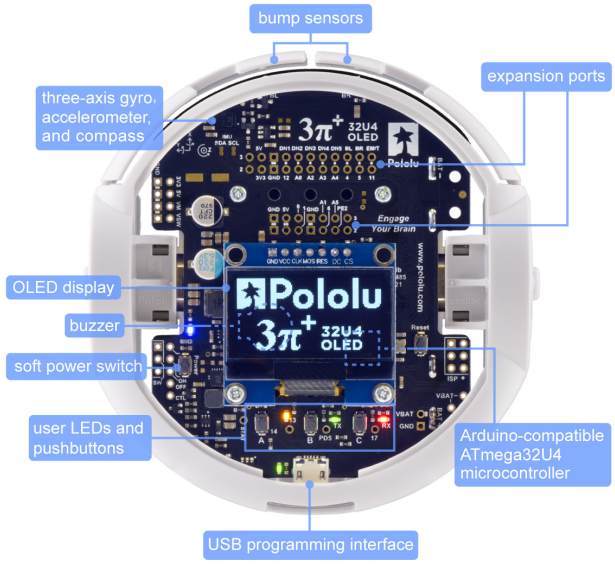

Fig. 1. 3 Pi+ Top View [2]

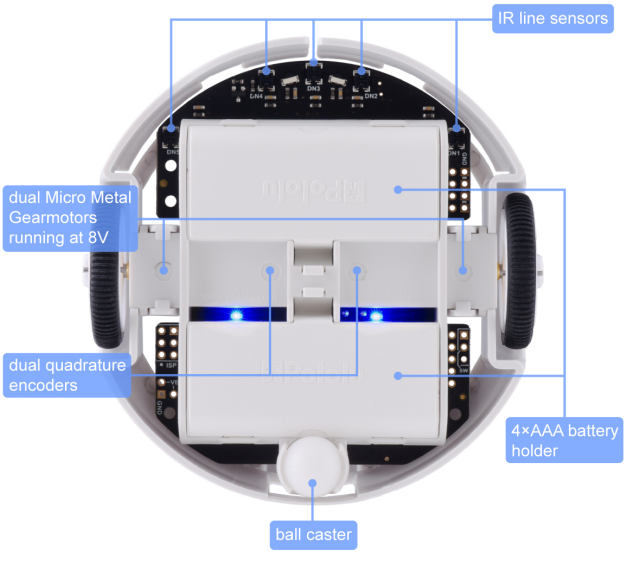

Fig. 2. 3 Pi+ Bottom View [2]

We have two limitations in our experiment. One is that the robot needs to be 2 cm away from the wall to avoid the wheels touching the wall or unnecessary collisions other than head-on collisions at the corner of the box. Another one is that the robot will not be able to calculate the exact perimeter of the square of the box as the distance travelled is calculated from the centre of the robot using kinematics calculation. Therefore, we need to add a constant of 4.5 cm to each side measurement as the length of the robot is 4.5 cm, which is the diameter of the Pololu 3Pi+ Robot [2].

A. Hypothesis Statement

The bump sensors, the rotary encoders and the kinematics are used to measure the test area and the bump sensors will cause errors when the robot hit the obstacle. Therefore, we hypothesise that:

> When the robot hits an obstacle in the test environment, it will cause an error in the rotary encoders and kinematics of the robot. We predict that the actual perimeter of our test environment will be close to the perimeter which we find using 3Pi+ robot after minimizing the errors accumulated in the encoders of the robot.

We investigate this hypothesis through a structured experiment on the Pololu 3Pi+ mobile robot, comparing the collision results using bump sensors with and without our technique.

II. Implementation

A. Preliminary Experiment

The preliminary experiments are necessary to understand at what speed the error in the encoder counts is minimum and whether the error in the encoder counts increases or not when the robot is travelling in a straight line. Also, we need to understand how the bumping is affecting the error in the encoder counts when the robot is travelling at different speeds. These preliminary experiments help us understand the base error that we have in the robot so that we can remove these base errors while calculating the perimeter of the test environment. See the baseline error and collision error in this section to understand the details of how these experiments are conducted.

We need some prerequisites for the robot to conduct preliminary and main experiments and they are:

- **PID**: The robot will have systematic errors in the wheels that avoids the robot to go in a straight line and we need to use a Proportional, Integral and Differential (PID) controller to make the robot travel in a straight line. As seen in Figure 3, we create a nested control PID, in which the Left and Right PID controllers are closed-loop control systems which are then controlled by a Heading PID to automate the demand velocity for the left and right wheels respectively and the heading PID tries to make the angle zero between the X and Y axis of the kinematics [3].
- **Bump sensors**: To detect the collision, we use the bump sensors of the robot. These are the sensors that are present on the front of the Pololu 3Pi+, they are photo-transistors which are present behind plastic flaps. We detect a bump by measuring the time taken by the photo-transistor to discharge the capacitor [2] as the sensor reading. The infrared(IR) emitter on the robot will make the capacitor charge and discharge quickly. One of the drawbacks of the robot is that we can't use both line sensors and bump sensors at the same time even though they work on the same principle. We need to turn on the IR to use the line sensors, whereas we need to turn it off to use bump sensors to get better readings from the sensors. If we attempt to use the bump sensors with the IR turned on then the robot finds it difficult to identify the bump as the sensor values are not going to change much from without bump phase to with bump phase. This is due to the distance between the IR sensors and the plastic flaps of bump sensors being very small in the robot. Another challenge we faced is that even on the same robot the bump sensor values differ from each other, in the graph shown in Figure 4 where the X-axis defines the time taken in microseconds for the capacitor to discharge and Y-axis defines the time which has elapsed. The peaks represent the value of the sensors when the plastic flaps in front of the bump sensors are completely depressed in which means that a collision has taken place. To tackle this issue, we need to calibrate both bump sensors and normalise them by dividing the sensor value by the highest value.
- **Bump sensors calibration**: If we look at Figure 4, we can identify that there is a bias in the sensor values due to which the readings are not starting from zero. We can make another observation that the value of each sensor differs from each other and this is due to systematic error. Calibration helps to remove these two errors. In our case, the bump sensor values are continuously calibrated using the below equation.

$$Sensor_I^{Normalized} = \frac{Sensor_I - bias}{Sensor_I^{max} - Sensor_I^{min}} \quad (1)$$

Here I is the sensor number, $Sensor_I^{Normalized}$ is the normalized value of the $Sensor_I$'s read value. $Sensor_I^{max}$ and $Sensor_I^{min}$ is the maximum and minimum value of the sensor respectively.

- **Kinematics**: The kinematics of the robot should be precise, as we want to make sure that the error does not vary based on our testing methods.

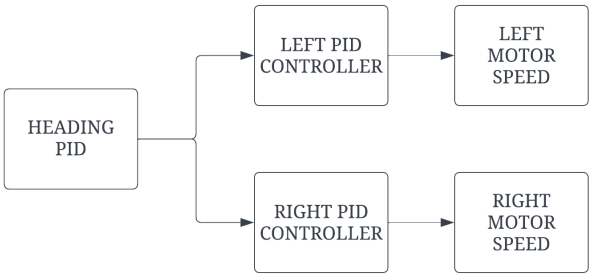

Fig. 3. PID Controller Block Diagram

To find out where exactly our robot is during its travel we make use of equations 2, 3, 4. Equation 2 defines the position of our robot on the X-axis at each time step "t", similarly equations 3 and **??** define the Y-axis position and the angle difference respectively.

$$X_I^{t+1} = X_I^t + (X_R \cos\theta_I^t) \tag{2}$$

$$Y_I^{t+1} = Y_I^t + (Y_R \sin\theta_I^t) \tag{3}$$

$$\theta_I^{t+1} = \theta_I^t + (\theta_R) \tag{4}$$

Here X_I and Y_I are the global frame co-ordinates.
X_R and Y_R are the local frame co-ordinates of the the robot at any point of time.
θ is the difference between the angle of the global frame and the local frame axes.

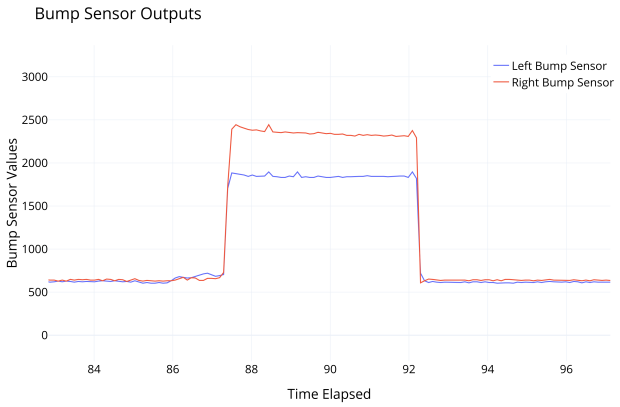

Fig. 4. Discharge values of capacitors

To understand the error in the odometry and kinematics, we conducted the following experiments:

- Baseline error test
- Collision error test

These tests are further elaborated below.

B. Baseline error

As an initial step, we conducted a simple experiment to understand how accurately the robot is able to track its position without any collisions. We used speed as an independent variable and understand how it is affecting the odometry or kinematics of the robot. The robot is allowed to travel a fixed distance of 100 mm at 3 different speeds (0.36 mm/s, 0.45 mm/s and 0.54 mm/s) to understand the error in the odometry. When the robot is allowed to travel a distance of 100 mm, the kinematics showed 2-3% at 0.36 mm/s, 3.5-4% at 0.45 mm/s and 6-6.5% at 0.54 mm/s extra travelled distance due to wheel slippage, when the robot is stopped suddenly. This can be observed through the graph shown in Figure 5. To reduce this error, we coded the robot to slow down its speed of travel when it reached close to our fixed distance of 100mm. As a result of this, the error in the kinematics reduced to 1-2 per cent compared to the robot not slowing down the experiment, this can be observed through the graph shown in Figure 6 and this error is independent of the speed as the robot is slowing when it reaches a fixed distance of 100 mm. From this, we have inferred that the robot may perform better when we use the IR sensors to detect the collision so that we can make the robot slow down when it reaches the collision instead of using bump sensors to detect the collision in our main experiment.

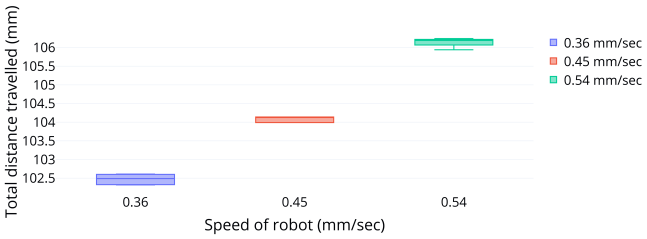

Fig. 5. Total distance travelled without slowing down before obstacle

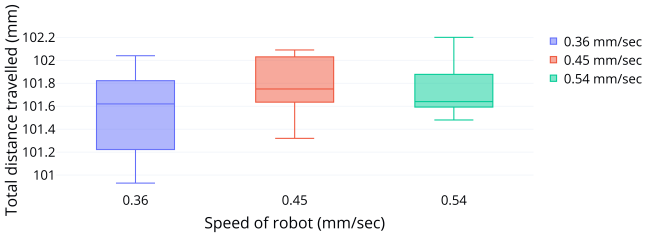

Fig. 6. Total distance travelled with slowing down before obstacle

C. Collision error

As our experiment is to measure the perimeter using bump detection, we need to understand how collision affects the error in the odometry. Also, as our test environment is the square section, we'll set the collision angle to 90°. we conducted two experiments in this section. One is that the robot travels a fixed distance of 200 mm and hit the obstacle and comes back to its original place. Another one is that the robot travels a fixed distance of 200 mm, but instead of hitting the obstacle at the same speed, the robot reduces its speed when it reaches the obstacle and comes back to its original place. We used speed as an independent variable and conducted the above 2 experiments at three different speeds, 0.36 mm/s, 0.45 mm/s and 0.54 mm/s to understand the effect of speed on the odometry error so that we can use better speed in our main experiment to measure the perimeter.

We observed that we conducted with the robot slowing down when it approaches the obstacle has an average odometry error of 19.5 mm from the starting point and the error is independent of the speed as the robot is slowing down when it approaches the target and the same is depicted in the Figure 10. Whereas, if the robot does not slow down before reaching the obstacle, the error in odometry increases with the increase in speed and it is shown in the Figure 9.

We observed that the strength of the collision is important to activate the bump sensors. If the target collision is not strong enough to push the bump sensor in either due to the speed of the collision not being enough or the obstacle is not heavy enough, then the bump sensors will not get activated. If the obstacle was not heavy enough, the robot would just keep moving forward while taking the obstacle

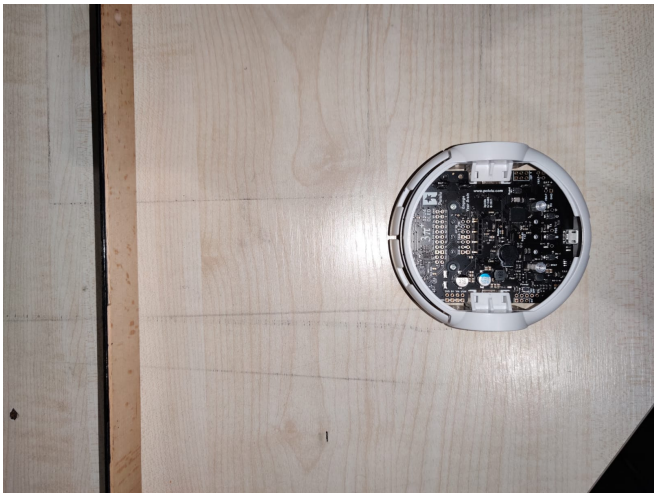

Fig. 7. Collision detection using Bump Sensors

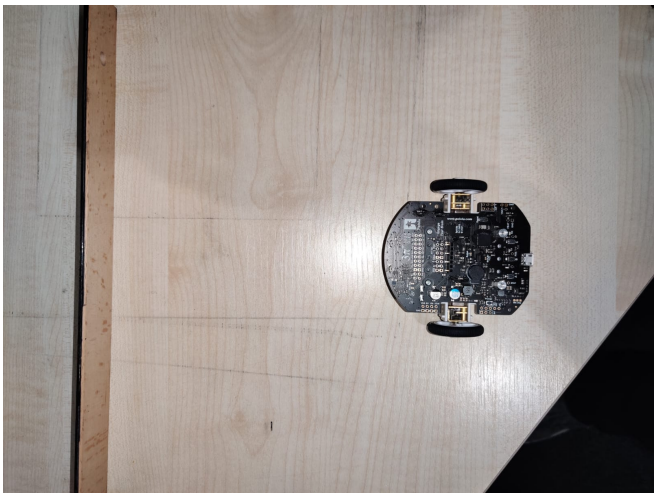

Fig. 8. Collision detection using IR Sensors

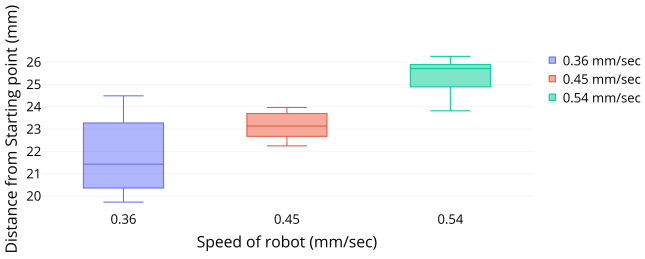

Fig. 9. Distance between start and end point of robot using Bump sensors

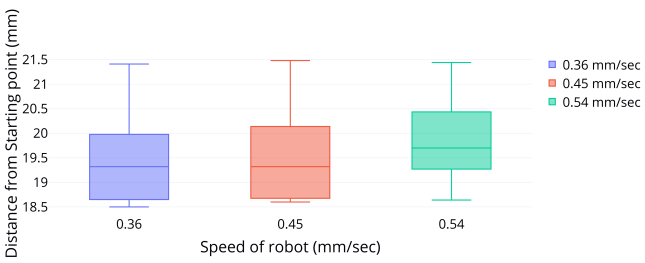

Fig. 10. Distance between start and end point of robot using IR sensors

with it. Therefore, we need to make sure that the walls of the test environment are strong enough to activate the bump sensors before we start the experiment.

As a consequence of these preliminary studies, we learnt that the technique of slowing the robot before it reaches the target position is better compared to not slowing down as we can observe that it has a lesser average error and is independent of the speed at which the robot travels.

III. Experiment Methodology

The objective of the experiment is to understand the error accumulated in the encoder counts or odometry while measuring the perimeter of the task environment. The robot can detect the collision using the bump sensors or the IR sensors but the robot will not be able to measure the angle of the collision to take the next turn. Choosing the square section will make the experiment simple as we can write the code to turn the robot 90 degrees whenever it detects a collision. Therefore, we've considered the square section of size 270*270 mm as our test environment to measure the perimeter. The robot will have total 8 motions, move forwards, turn right, move forwards, turn right, move forwards, turn right, move forwards and stop, in the experiment to measure the perimeter of the box.

A. Overview of Method

We have learnt from the preliminary experiments that our robot will have less error in the odometry when the robot slows down when it approaches the collision compared to the robot hitting the collision without slowing down when it approaches the target. Therefore, in our experiment to measure the perimeter of the test environment, we considered running the robot around the edges of the test environment while detecting the obstacles using IR sensors. Also, to validate the preliminary results, we run the robot along the edges of the test environment while detecting the collision using bump sensors to better understand the odometry error difference in both methods. Both experiments involved the following steps:

- We chose the brown cartoon box of size 270mm * 270mm as the edges of the box are good enough to activate the bump sensors.
- We mentioned the limitations of our work in the introduction. We need to place the robot 20 mm away from the wall to avoid unnecessary collisions with the walls. Place the robot at 4 corners of the test environment in such a way that it goes in a straight line or as mentioned in Figure 11.
- Turn on the robot and step back.
- The robot will travel along the edges of the test environment and comes back to the original place and the robot will give us the distance travelled at each edge. The same task is performed times for three different speeds, 0.36 mm/s, 0.45 mm/s and 0.54 mm/s for both experiments to evaluate the average error and to understand the effect of speed on the odometry error.

B. Discussion of Variables

There are several variables in the experiment we need to control to complete the task at hand and that makes

the robot less intelligent. Both controlled and uncontrolled variables are explained in detail below.

- **Controlled Variables**:
  - Ambient Light Conditions: The normalised/calibrated values from the bump sensors are dependent on the light conditions of the task environment. We need to make sure that the normalised values of the bump sensors work correctly before we start the experiment as mentioned in the introduction/implementation.
  - Speed of the robot: The robot is allowed to run at three different speeds 0.36 mm/s, 0.45 mm/s and 0.54 mm/s and the speed in each iteration remains the same as it affects the slippage of the wheels. When we are using the IR sensors to detect the collision, the speed of the robot reduces as it approaches the collision and increases as it moves away from the collision.
  - Position and angle of the starting point: The robot should be placed correctly so that it goes parallel to the wall and in a straight direction as mentioned at the beginning of this section, otherwise the perimeter calculated would be affected.
  - Collision angle: As the task environment is in a square shape, the collision angle is always 90°. If the collision angle is other than 90°, then the perimeter calculation might be affected as the robot won't be measure the angle of the collision and it only takes 90° turn.
  - Surface friction: We made sure that the surface is clean so that the robot can run smoothly on the surface before starting the experiment. We used the surface of the study table in our experiment on which the robot was running smoothly.
  - Battery level: To avoid errors due to speed variation or the halting of the robot due to low batteries, we used fully charged batteries while conducting the experiment.
- **Independent Variable**: We are using the speed as an independent variable to measure the perimeter of the task environment multiple times with different speeds,0.36mm/s, 0.45 mm/s and 0.54 mm/s to see the difference in the odometry error. Varying the speed of the robot helped us to understand the error in the odometry of the robot.
- **Dependent Variable(s)**: The dependent variable is the perimeter of our test environment, the square box. We used the below equation **??** in the code to measure the perimeter of each side and each side is named as X1, X2, X3 and X4, and the added them to get the perimeter of the test environment.

$$count_{total} = \frac{count_{LeftEncoder} + count_{RightEncoder}}{2} \tag{5}$$

Here $count_{total}$ is distance travelled in encoder counts. $count_{LeftEncoder}$ and $count_{RightEncoder}$ is the encoder count of Left and Right Wheel respectively.

Fig. 11. Test Box

$$perimeter = count_{total} \times \frac{Wheel_{Circumference}}{GearRatio} \tag{6}$$

Here $Wheel_{Circumference}$ is the circumference of the wheel of the robot and $GearRatio$ is the Encoder counts per rotation of wheel. We take the value of this as 0.28 mm/encoder counts. This makes the equation as 7, which gives us the distance travelled in mm.

$$perimeter = count_{total} \times 0.28 \tag{7}$$

C. Discussion of Metric(s)

The objective of the experiment is to measure the error produced while traversing the test environment and this section gives a good understanding of how the experiment was evaluated. The following metrics are used to measure the error in the experiment.

1) Mean and Variance: The mean (Equation: 8) gives the average perimeter of the test environment as the robot is measuring it. This was helpful to understand the impact of speed and bumping on the accuracy of the test area measurement. The out-liners in the iterations can heavily affect the mean a lot and might misguide us in understanding the error, but this metric is worth considering as it will give a good understanding of the errors if we don't have any outliners. The standard deviation (Equation: 9) gives the variation of the values and helps us to understand how the measurements are distributed and remove the outliners.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{8}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}} \tag{9}$$

Here μ is Mean, N is the size of population, x_i is each value from the population and σ is the Standard Deviation.

2) Ground truth: We measure the actual perimeter of the box using a scale. The perimeter measured by the robot is compared against the ground truth to understand the percentage of error in the perimeter. This gives us an idea of how accurate the robot is and helps us to reduce the error using different methods.

IV. Results

The experiment was conducted to measure the perimeter of the square test environment, which can eventually be used to calculate the area of the environment. The experiment was conducted in two ways, one with bump sensors and the other with IR sensors, the data was collected for every 5 iterations at three different speeds (0.36 mm/sec, 0.45 mm/sec, 0.54 mm/sec). Even though the size of the test environment is 27* 27 cm, the area to be measured by the robot will be reduced to 16*16 cm due to two reasons. One is due to the size of the robot, it cannot go to the ends of each corner and the perimeter is measured from the centre point of the robot. The diameter of the robot is measured to be 9 cm [2], because of this we have to reduce 4.5 cm from each side of the box from the ground truth to compare the results with the ground truth. We also noticed that if the robot is placed too close to the corner of the environment the wheels would either brush against the wall and stop the robot from travelling in a straight line or touch the walls during the turn and go off course. This was tackled by leaving a space of 2 cm from the edge of the environment which ensured that the wheels never touch the walls during the entire movement of the robot.

A. Collision with bumpsensors

Initially, the robot was programmed in such a way that the robot will move in a straight line and parallel to the edge of the environment till it reaches the target collision. Once the collision is made, the robot will take a right turn and then travel in a straight line parallel to the edge of the environment. The same process continues till it reaches the starting point. The data was collected for all four sides of the environment and at three different speeds (0.36mm/s, 0.45 mm/s and 0.54mm/s) and the results are mentioned in Figure 12. The mean of the perimeter of the measured environment is approximately 680 mm and this mean increases as we increase the speed of the robot, we can also note that the average standard deviation of all 3 speeds is ± 4.6 and the same has been mentioned in the Figure 12.

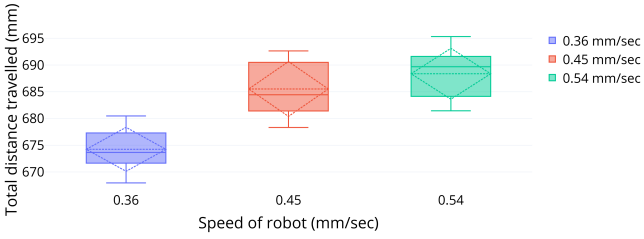

Fig. 12. Perimeter of the box with Bump sensors

B. Collision detection using IR sensors

To reduce the error, we removed the cover of the Pololu 3Pi+ robot, this results in the removal of the plastic flaps in front of the IR sensors used for the detection of bumps, due to this we can use the IR sensors directly to detect collisions. The experiment was programmed in such a way that the robot was able to detect the collision before it hits and slow down the speed of the wheels to reduce the encoder errors. As seen in Figure 9 and 12 we can observe that the slippage of wheels causes an error in the encoder counts which increases with an increase in the travelling speed of the robot, however, in Figure 10 and 13 we can observe that the encoder count errors do not increase with the increase in speed, with a constant average error in the range 1.5-3.0 mm. The mean of the perimeter of the measured environment is approximately 660 mm and the average standard deviation of all 3 speeds is ± 7.64. This doesn't increase with the speed as the robot slows down when it approaches the target.

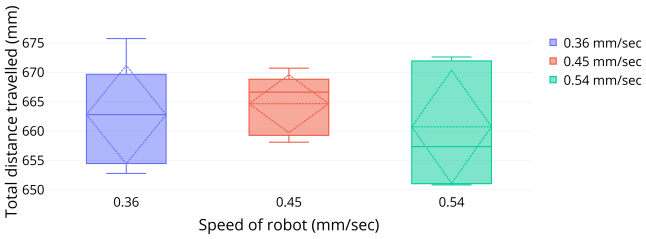

Fig. 13. Perimeter of the box with IR sensors

C. Comparison

From the graphs present in Figure 12 and 13, we can deduce that the average perimeter of the box with bump sensors and IR sensors was found to be 682.731 mm and 662.738 mm respectively. If we compare this with our actual measured perimeter which is 660 mm, we can see that there is a reduction of 3% odometry error when using IR sensors to detect the collisions. This validates our preliminary experimental results that a jerky stop when a collision occurs results in an increase in total error accumulated by the system.

V. Discussion and Conclusion

Our hypothesis was

> When the robot hits an obstacle in the test environment, it will cause an error in the rotary encoders and kinematics of the robot.
> We predict the area of the test environment close to the ground truth minimizing the error in the encoders and the kinematics.

We have succeeded in reducing odometry error by adopting a technique of reducing the speed of the robot when it approaches the obstacle by use of IR sensors instead of bump sensors. Even though this technique reduced the error, it was not able to remove the error completely and this could be because of the random error. We think that further the same experiment can be evaluated using different shapes of the test environment with or without using the other sensors in the robot and analyse how the error varies to understand the robotic system in detail.

REFERENCES

[1] iRobot, "irobot® roomba 500 open interface (oi) specification," in *iRobot® Roomba 500 Open Interface (OI) Specification*, pp. 1–35.

[2] Polulu, "Pololu 3pi robot user's guide," in *Pololu 3pi Robot User's Guide*, pp. 1–57, 2001-2009.

[3] Paul, "Github - robotic systems: Ematm0054$_2$2 $-$ 23,$''$ 2022.