Obstacle Avoidance using a Single IR TOF Sensor

James Pegg - jp17245 Aidan Higgins - ah17712

***Abstract*—We report the development of a line following AGV system with an implemented obstacle avoidance algorithm using the minimum number of sensors is discussed. Discussion and plots demonstrating successful line following and kinematics, calibration of odometry parameters and initial successful obstacle avoidance is provided. An error metric used to compare performance of obstacle avoidance algorithm operating with different parameter conditions is developed with data and plots to visualise the affects of key algorithm variables on avoidance accuracy and speed. A distance offset value of $d_{offset} = 140$ mm and jump threshold of $J = 20$ mm are recommended for the algorithm developed.**

I. Introduction

This report explores the capabilities of an obstacle avoidance system using a single infrared (IR) Time-of-Flight (TOF) proximity sensor on an automated guided vehicle (AGV). The implemented system aims to be capable of following a arbitrary line path, similar to Assignment 1, whilst making use of a TOF proximity sensor to detect and avoid an object blocking the route. Upon detection of an obstacle the system should be able to traverse around the perimeter of the object before rejoining the line path. The experiment investigates the effects of altering key obstacle avoidance algorithm parameters on the systems accuracy and speed, as defined below.

AGVs are commonplace within industrial factories and their utilisation continues to rise with the market predicted to exceed $2.74 billion by 2023 [1]. AGVs were first introduced into factory settings in 2005 and functioned by following a predetermined route laid out using magnetic tape [1]. These systems are restricted to fixed routes and are capable of detecting obstacles but are unable to navigate around them [2]. The vehicles therefore come to a halt upon detection of an obstacle and are idle until the obstacle is manually moved; this can incur large delays and requires human intervention. This report explores the potential in improving the design of AGV's by allowing for the navigation around obstacles to be undertaken before rejoining the predetermined route. The implemented system aims to be able to provide a robust, accurate and timely obstacle avoidance algorithm, using a single sub $9 TOF proximity sensor [5]. This algorithm presents a solution using the minimum equipment required for obstacle avoidance, or as a redundant method for when sensors fail in hardware platforms with multiple sensor configurations.

II. Hypothesis

"We hypothesise that when using a single TOF proximity sensor for obstacle avoidance the accuracy can be improved by reducing the proximity to the obstacle whilst traversing the perimeter. This is due to the reduced precision of the TOF sensor at greater distances, which will lead to an increase in error for the obstacle corner detection & calculations. We also hypothesise that the operating speed of the obstacle avoidance algorithm will be improved with a greater distance from the obstacle. This will be as a consequence of the reduced rotation required by the Romi to scan the same length edge of an obstacle when at a further distance. It should be possible to construct a combined error metric that takes into account the time taken and the distance error between the Romi and obstacle perimeter; for which we predict there to be a minima."

III. Hardware

This system utilises the Romi chassis along with the 32U4 Control Board by Pololu, which allows for a programmable and customisable system design. This is integrated with the Romi Encoder Pair Kit allowing for closed loop motor control [3]. The 32U4 Control Board is preloaded with an Arduino-compatible boot-loader [3] allowing for easier system implementation using the Arduino IDE.

In order for the Romi to be capable of following a line path, laid out using black tape, a 3 channel QTR-MD-03A Reflectance Sensor Array is used. This array of IR LED and phototransistor pairs allows for the accurate detection of changes in reflectance [4], and is positioned underneath the chassis directed downwards towards the floor. This sensor allows the Romi to distinguish between background flooring and a black tape line.

The overall objective of this experiment is to explore the potential for an obstacle avoidance system utilising a single TOF sensor. The Romi makes use of the Pololu VL6180X TOF distance sensor [5]. The sensor allows for ranging measurements to be made up to 200 mm, or up to 600 mm with a reduced resolution [5], by making use of infrared pulses. These pulses allow an accurate distance measurement to be determined irrespective the colour or surface of a target [5]. The VL6180X sensor is mounted parallel to the floor on the front of the Romi chassis, as indicated in Fig 1, and allows for obstacles at a distance in front to be detected.

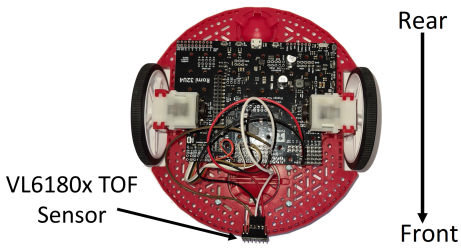


Fig. 1: Hardware configuration used in this experiment

IV. Line Following & Odometry

A. Sensor Readings

The QTR-MD-03A Reflectance Sensor Array is placed on the underside of the Romi chassis and allows for the reflectance of the ground to be measured. A strongly reflected signal pulls the analog voltage output of the sensor towards ground with a poorly reflected signal pulling it towards $V_{in} = 5$ V. These values are mapped to a digital value range of 0-1023 using the Arduino internal ADCs.

The raw sensor readings need to be calibrated to take into account changes in ambient light conditions and floor reflectances when the Romi is used at different time or places. The Romi starts on the normal ground surface and takes an average of 10 sensor readings to calculate the sensor bias. This bias is subtracted from all future readings. The calibrated reading is normalised using a user entered sensor gain value for each of the three sensors $i_c = (i_{raw} - \bar{i}_{raw})/k$. The final sensor reading varies as a float from 0 to 1 when on the ground surface to on the black tape line as shown in Fig 2.

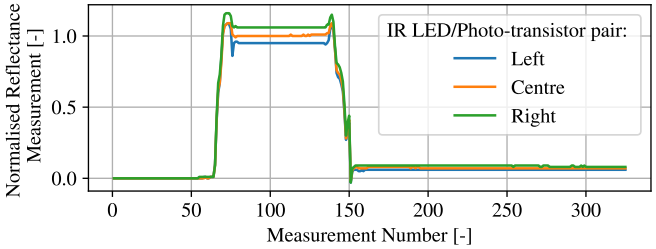


Fig. 2: Normalised line following sensor readings

B. Line Following

The line path is laid out on the flooring using black electrical tape with a width of 18 mm. Initially the Romi is configured to drive forwards until the black line is detected. Upon detection the Romi starts to follow the line. The normalised sensor readings are used to set each motor velocity in order for the Romi to continue to lie directly above the line whilst travelling around curves and corners.

If the Romi arrives at a sharp turn in the path then the three sensors may all stop detecting the line simultaneously. At this point the Romi will scan 90 degrees in both directions in an attempt to locate the new direction of the line. If at any point during the scanning a black line is detected the Romi will stop turning and continue to follow the line path as mentioned above. If after scanning both left and right no line is discovered it is assumed that there is a gap in the line path and the Romi will drive forwards from its last position until it reaches the black line. The line following algorithm is demonstrated successfully in Fig 3a.

An additional Boolean line detected flag is provided internally for use with other methods to indicate when the black line is detected. A threshold value of 0.4 is used for each line senor.

C. Movement & Odometry

The motion of the Romi is governed by a PID (Proportional, Integral, Derivative) controller for each motor which aims to keep the wheel velocity equal to each saved target velocity. The target velocity for each motor can be set from a variety of different internal sources. The motor PID controllers allow for the Romi to adapt to the environment ensuring each wheel continues to rotate at the desired speed regardless of wheel resistance (e.g. surface or incline) assuming a non-slip condition. Differences between motor gear boxes and other manufacturing deviations are mitigated through the use of a velocity PID controller.

To determine the wheel velocity the rotation of the motor shaft can be monitored using a magnet mounted on the shaft and two hall effect sensors to create a quadrature encoding of the shaft position [6]. The quadrature encoding is used to keep track of the rotation of the Romi wheels which can then, through odometry [7], allow the Romi to keep track of its position and rotation; the Romi's Pose. An interrupt service routine, triggered by the internal Timer3 timer, is used to periodically update the motor PID controllers, update the odometry and calculate the wheel velocities. The origin of the global X-Y axis is initialised at the Romi's current position upon start-up with the positive x axis as forwards and the angle, θ, defined from positive X axis counterclockwise.

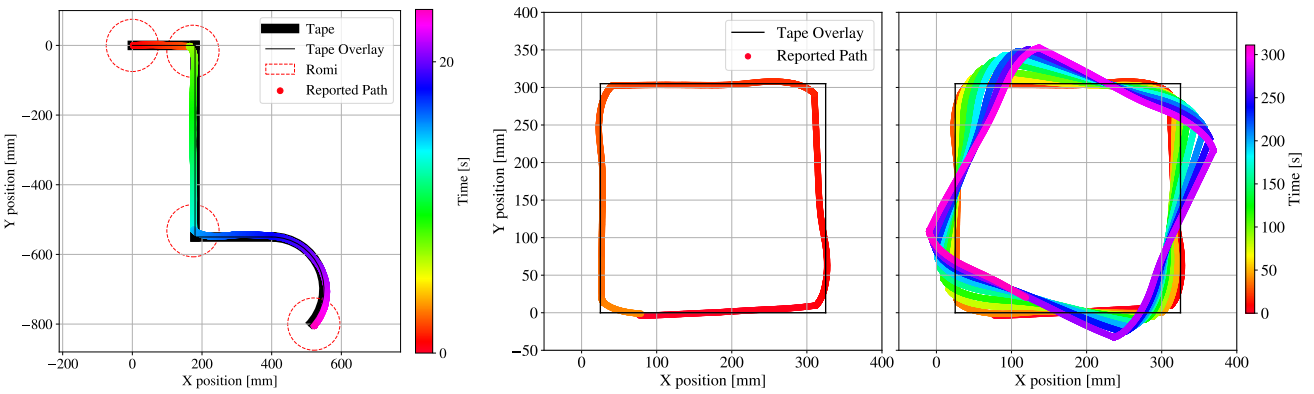
The odometry is used by the Romi to keep tack of its position during line following and during obstacle avoidance. Two additional PID controllers are used to control the Romi movement when not line following. The first PID controller allows for the Romi to point towards an arbitrary point in the XY global coordinate frame and the second PID controller allows the Romi to travel straight to the point after first rotating to face it. High accuracy for the odometry is essential for obstacle avoidance to allow for the Romi to navigate to the correct location to avoid a collision. Fig 3a demonstrates the capabilities of the Romi to traverse along a predefined route, along with the Romi's estimated position generated using the on-board odometry.

Fig 3b shows the Romi traversing along a square track. The coloured line indicates the reported position of the Romi. This was done to validate the functionality of the odometry and used to calibrate key parameters used in the odometry; the wheel base of the Romi W and the encoder pips per mm of linear travel P. The rotation calculated by the odometry is impacted by both the W and P values whilst the forward distance is only affected by the P value. With this in mind the odometry can be calibrated to provide a higher accuracy between the Romi reported position and the real world position during extended run times. Fig 3b shows an accumulation of rotational error after 5 minutes of continuous traversal after manual calibration of the odometry parameters. This indicates that the wheel base parameter W is slightly incorrect however the experiments performed for this report were much shorter in duration and so this error was deemed acceptable.

V. Obstacle Avoidance

A. Algorithm Definition

The implementation of the obstacle avoidance algorithm uses a single VL6180X TOF sensor. This sensor is capable of detecting an obstacle within proximity, allowing the Romi



(a) Romi following path

(b) Romi following square path, illustrates odometry error accumulation

Fig. 3: Romi odometry plots

to act accordingly and calculate the necessary coordinates in order to traverse the perimeter of an obstacle to reach the line on the other side.

The Romi is programmed to autonomously follow a path laid out using black tape as discussed in Section IV. Following this the Romi will continuously monitor the reading from the VL6180X sensor. If at any point the sensor reading falls below a set value the Romi will come to a stop and flag that an obstacle blocking the path has been detected. The Romi will rotate counterclockwise whilst monitoring the reading from the VL6180X sensor. A corner is detected when there is a large change between two consecutive sensor readings as the sensor goes from detecting the object to detecting the background behind (or saturating at the maximum range). Using the Romi odometry the x and y coordinates of the obstacle corner can be calculated using (2). To avoid a collision between the obstacle and the Romi, an offset of the corner coordinates must be calculated using (3). Once the offset point has been calculated the Romi will drive to these coordinates, before rotating to face the corner it previously detected. Fig 4 shows an overview of the obstacle avoidance algorithm.

$$d_{corner} = d_{raw} + D \tag{1}$$

$$\begin{bmatrix} x_{corner} \\ y_{corner} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + d_{corner} \times \begin{bmatrix} cos(\theta) \\ sin(\theta) \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} x_{offset} \\ y_{offset} \end{bmatrix} = \begin{bmatrix} x_{corner} \\ y_{corner} \end{bmatrix} + d_{offset} \times \begin{bmatrix} cos(\theta + \pi/2) \\ sin(\theta + \pi/2) \end{bmatrix} \tag{3}$$

This process is repeated with the Romi rotating counterclockwise to scan for the next corner of the obstacle. At any point whilst travelling to corner offset point the Romi passes over an black line then the Romi will go back to line following. Line detection is disabled when travelling to the first corner of the obstacle to avoid erroneously detecting the line segment the Romi has just departed from. A flowchart, Appendix A, shows an overview of the workings of both the line following and the obstacle avoidance. Algorithm 1 provides the obstacle avoidance pseudo-code.

B. Time-Of-Flight Sensor

A single VL6180X TOF sensor is used to measure the distance between the Romi and an obstacle ahead. The VL6180X

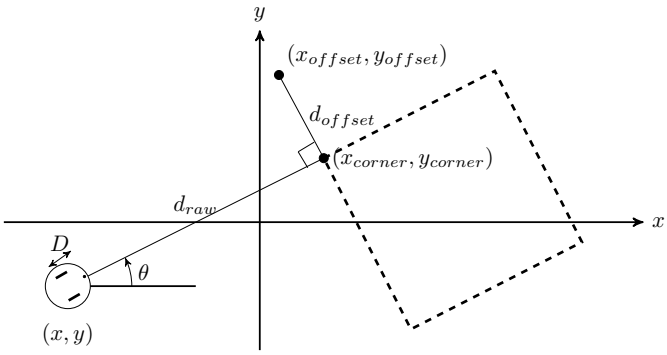


Fig. 4: Obstacle avoidance variable definitions

Symbol	Definitions
d_{corner}	Distance to obstacle corner from Romi centre
d_{raw}	Distance to obstacle corner from sensor
D	mm from Romi centre to the VL6180X, constant 85mm
x	Current x coordinate of the Romi relative to global axis
y	Current y coordinate of the Romi relative to global axis
θ	Angle from the global x axis to the obstacle's corner
x_{corner}	x coordinate of the obstacle relative to global axis
y_{corner}	y coordinate of the obstacle relative to global axis
x_{offset}	x coordinate of the corner offset
y_{offset}	y coordinate of the corner offset
d_{offset}	Distance between obstacle corner and offset coordinates

Table I: Definitions of symbols used in equations 1-3

Algorithm 1 Avoidance

```
procedure ObstacleAvoidance
    distance_threshold ← 200.0 − 85.0
    Line_Following :
    Follow Line
    if distance_to_obstacle > distance_threshold then
        Find_Corner :
        Rotate Left
        if CornerDetected then
            Calculate Corner Position
            Calculate Romi Offset Position
            Drive to Offset Position
            if LineDetected then
                goto Line_Following
            Point at Corner position
            goto Find_Corner
```

measures the time taken for an emitted infrared light pulse to reach the object and reflect back to the sensor. Using this time and the known constant speed of light the distance between the sensor and an object can be calculated. Table II shows the scaling factors available with the compromise of range and resolution visible [5].

Scaling factor	Max Range [mm]	Resolution [mm]
1	200	1
2	400	2
3	600	3

Table II: VL6180X TOF Sensor scaling factors

Appendix B shows the suggested accuracy of the device supplied by the manufacturer. As can be seen there is a linear relationship between the measured distance and actual distance up to to 150 mm. The accuracy of the supplied device was verified in order to check it follows the expected trend before obstacle avoidance experiments were undertaken. Fig 5 shows the measurements acquired from the supplied TOF sensor when measuring the distance to a piece of plain white paper. As can be seen the sensor is able to provide accurate measurements demonstrating the same linear relationship as the supplied specification up to to 150 mm. Fig 5 shows sharp increases in the measured distances at 200 mm for scaling 1, 400 mm for scaling 2 and 550 mm for scaling 3. These sharp increases occur due to the saturation of the sensor and the device is unable to achieve the theoretical maximum of 600 mm as mentioned in the specification. The accuracy for both scaling 2 and 3 starts to decrease beyond 200 mm with the sensor measuring distances with 10 % error from $\geq$ 350 mm.

Within this experiment a scale factor of 3 is used to allow for the increased range when scanning along the edge of an obstacle. It is often the case that the sensor will saturate when attempting to detect the corner of an obstacle as it falls beyond the sensor's range. In order to overcome this the Romi is able to traverse part way along the obstacle edge and scan for the corner again as illustrated in Fig 8.

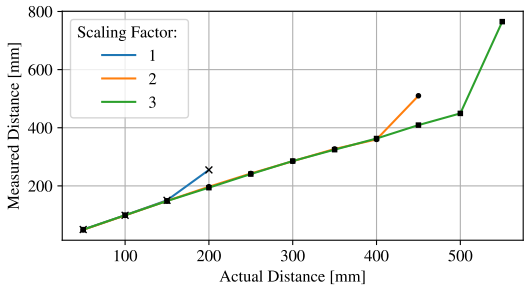


Fig. 5: Actual distance vs TOF sensor measurements, full data in Appendix C

C. Variables, Obstacle Characterisation & Initial Verification

In this experiment the obstacle geometries have been limited to rectangular cross sections but we believe the obstacle avoidance algorithm developed should be able to function with any convex polygon with internal angles greater than 90 degrees. Fig 6 details all the variables required by the algorithm to be able to effectively calculate and traverse the perimeter of an obstacle.

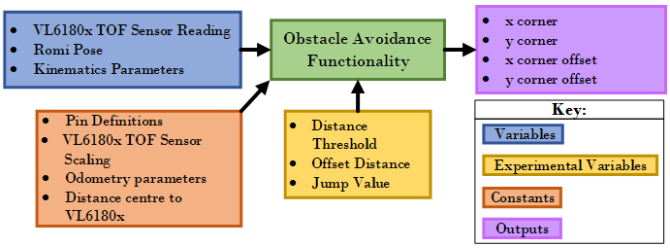


Fig. 6: Obstacle avoidance variables and constants

Fig 7 illustrates the successful avoidance of an obstacle which blocks the same path as in Fig 3a. This implementation is able to accurately avoid the obstacle whilst incurring a small delay, of roughly 30 seconds, from the start to finish of the tape path.

D. Error metric definition

The success of the obstacle avoidance algorithm developed has been demonstrated. In order to evaluate the algorithm and as a metric for comparison between different operating conditions an error metric capable of taking into account the speed and accuracy of the algorithm is essential. The error metric chosen is the time integral of the distance error from the Romi to the obstacle perimeter, as defined by equation 4. The time component of the error metric is important as the system needs to be able to avoid a collision whilst reducing the time penalty of deviating from the path, especially in time sensitive environments. The distance error is important as the Romi should deviate from the path by the smallest amount possible; this is to avoid encroaching on other AGV paths and provide more predictable behaviour.

$$\epsilon = \sum_{n=1}^{N} (t_n - t_{n-1}) \times min\left(|\vec{d}|\right) \tag{4}$$

Where ϵ is the error metric, t_n and t_{n-1} are the time steps between consecutive points and N is the total number of data points. $\vec{d}$ is the distance between the Romi and the obstacle perimeter. The error metric is calculated using a Python script operating with saved data received from the Romi and a user defined obstacle perimeter. The error metric will have units of metre seconds [ms].

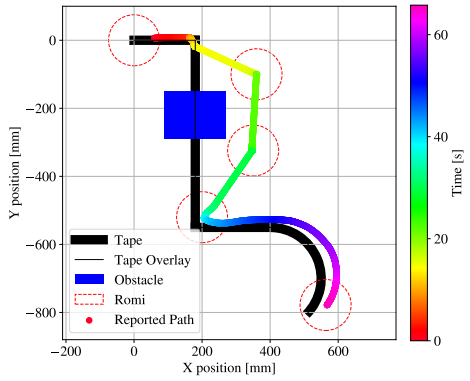


Fig. 7: Plot of path taken by Romi when following line with an obstacle in the way

VI. Results

A. Methodology & Intial Results

The Romi was set to follow a straight line path with an obstacle placed at the end. The Romi will detect the obstacle, navigate around the obstacle and halt immediately upon rejoining the initial line. Custom serial print functions were programmed into the Romi to allow for easy reporting back to the host PC. The host PC runs a Python script capable of parsing the incoming serial data and will dynamically generate a dictionary that stores all variable data. The reported position of the Romi, calculated from the on-board odometry, was taken to be entirely accurate, this represents a drawback of the experiment method currently used. The obstacle geometry and position relative to the start of the line, the Romi origin coordinate, is manually entered into the Python script which can then calculate the minimum distance between all Romi pose coordinates and the obstacle perimeter. The time integral is performed using the timestamps attached to each data point upon receiving the data.

The obstacle was a rectangular box with dimensions of 310 x 190 mm. Fig 8 shows the Romi avoiding the obstacle and halting when rejoining the line. This right plot shows the distance error for each data point and the positions of the Romi at a corner detection events along with the corresponding detected corner (or where sensor saturation has occurred).

The obstacle avoidance algorithm variables that are investigated are the offset distance from the corner at which the Romi will travel to, and the jump threshold value used to compare consecutive TOF sensor readings when detecting a corner. The range of values explored are provided in Table III. In order to minimise the number of experiments required to be run the distance threshold, the distance between the obstacle and the Romi before the Romi 'detects' the obstacle, was set to be equal to the offset distance for all experiments. Larger offset distances were not tested due to the limited space available for running experiments. The smallest possible value would be equal to the Romi diameter at 75 mm but obstacle collision would be highly probable.

Parameter	Symbol	Units	Min	Default	Max
Offset distance	d_{offset}	mm	100	140	220
Jump threshold	J	mm	8	20	60

Table III: Experiment parameters

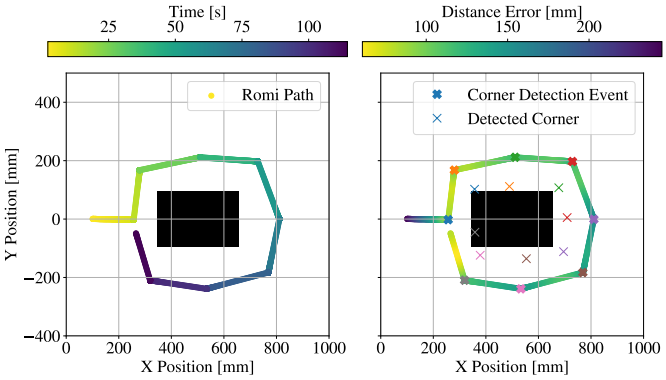


Fig. 8: Romi position during obstacle avoidance

Running the experiment with the initial parameters of $d_{offset} = 140$ mm and $J = 20$ mm yielded an error metric value of 30781 metre seconds. This served as a baseline value with the affects of increasing and decreasing the parameters explored below.

B. Varying Target Obstacle Offset Distance

Fig 9 shows the affect of varying the target obstacle offset distance for a variety of metrics with a fixed jump threshold value of 20. The experiment for each offset distance was repeated three times with the average, minimum and maximum values indicated by the black error bars. Additional plots are provided in Appendix D.

The error metric appears linearly proportional to the offset distance where a reduction in offset distance from 220 mm down to 100 mm yields a 49 % reduction in the error metric.

The average distance offset is always greater than the target offset distance, ideally the two values should be equal, but at lower target offset distances the average offset is smaller. This indicates the system is more accurate when travelling closer to the obstacle perimeter.

The time taken to navigate around the obstacle is predominately due to the number of corner detection events that the Romi has carried out. At offset distances of 180 and 210 mm the Romi navigated around the obstacle with 7 events twice and with 8 events once. These additional events required an additional 10 seconds. Once the number of corner detection events is taken into account, the total time is proportional to the offset distances, with smaller offset distances requiring slightly less time than larger ones. There is only 3.5 % reduction from an offset distance of 160 mm down to 100 mm.

C. Varying Jump Threshold

Fig 10 shows the affect of varying the jump threshold value J, used when detecting a corner, for a fixed offset distance value of $d_{offset} = 140$ mm. Additional plots are provided in Appendix E.

The error metric does not change considerably for different J values with a maximum difference of 7 % shown here from a threshold value of 40 mm to 8 mm. There is no noticeable trend.

The average distance offset decreases significantly with smaller J values following a logarithmic trendline. With a jump threshold value of $J = 8$ mm the average distance offset is 149 mm, which is only 9 mm greater than the target offset distance. At larger J values the average distance offset increases to > 180 mm showing a decrease in system accuracy.

The number of corner detection cycles increases as the J value is decreased which causes a large increase to the time taken to navigate the obstacle. Using the extreme values shown in Fig 10 the total time taken can increase by as much as 30 % by using smaller values of J.

D. Discussion

Varying the offset distance d_{offset} has a linearly proportionate affect on the error metric devised in this project.

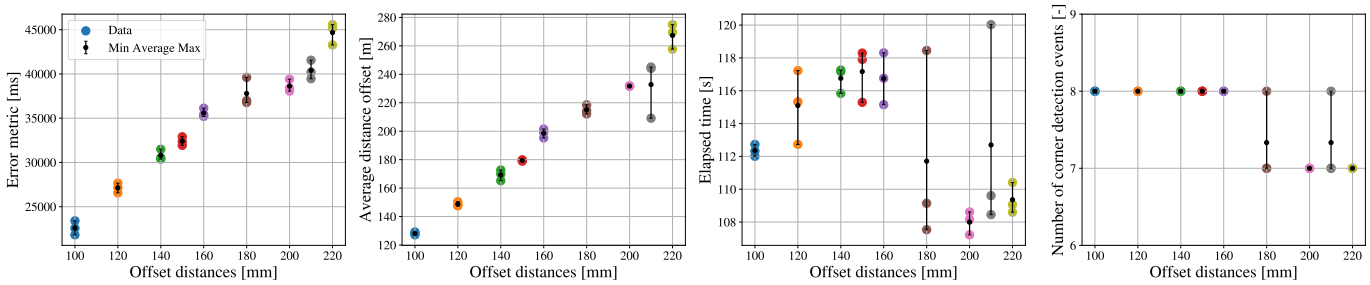


Fig. 9: Results of varying target offset distance d_{offset} with fixed jump threshold $J = 20.0$ mm

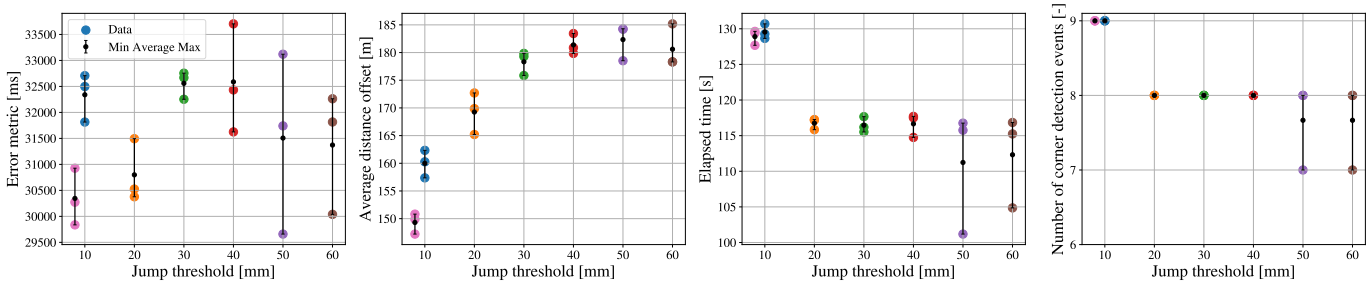


Fig. 10: Results of varying jump threshold J with fixed target offset distance $d_{offset} = 140.0$ mm

Varying the jump threshold J does not greatly affect the error metric. Both d_{offset} and J affect the average distance offset of the Romi to the obstacle perimeter and thus affect the systems 'accuracy'. Minimising the number of corner detection events is key in reducing the time spent navigating around the obstacle. Smaller offset distances provide more accurate tracking and a reduced error metric but slightly longer traversal time.

Varying jump threshold does not have considerable affect on the error metric but provides a controllable variable for sacrificing system speed over accuracy or vice versa. This error metric is arbitrary and could be altered depending on the relative importance of speed vs accuracy which will depend on the final system use case. The obstacle avoidance algorithm developed is highly robust with few errors encountered during the gathering of the data shown (a total of 42 experiments) but the Romi was more likely to collide with the obstacle with very small offset distances, $d_{offset} \leq 110$ mm.

VII. Conclusion

A. Conclusion

The system developed is capable of completing the line following task of Assignment 1 whilst also now being able to traverse around obstacles. The recommended operating parameters for the current algorithm are $d_{offset} = 120$ mm and $J = 20$ mm to provide good compromise between system accuracy and speed whilst maintaining high robustness.

Through gathering experimental data we can confirm, against our hypothesis, that when using a smaller target offset distance the Romi will traverse more accurately around the obstacle. At larger offset distances the Romi will traverse faster around the obstacle but due to requiring less corner detection events and not due to faster edge scanning as hypothesised. The time taken is predominately due to the number of corner detection events where larger offset distances can yield less corner detection events and thus take less time. We devised an error metric capable of providing a single value to describe the performance of the obstacle avoidance algorithm operating with a given a set of parameters. This error metric was shown to decrease by up to 49 % when travelling closer to the obstacle perimeter but for varying offset distances there does not exist a minima.

B. Further Work

To further develop the system continued experimentation on additional obstacles with different sizes or shapes should be undertaken to verify the reliability of the system. Any errors or unforeseen behaviour should subsequently be addressed.

The current algorithm makes use of a single TOF proximity sensor, which in turn means that the Romi must rotate in order to scan for the corner of an obstacle. The addition of multiple TOF sensors mounted as an array on the front of the Romi would allow for a larger area to be detected and therefore may reduce the need for rotation as the corner as been detected by one of the outer sensors.

The experimentation results are currently reliant upon the Romi odometry which is taken to be entirely correct, which is not the case as shown in Fig 3b. Use of an external tracking system to provide Romi position data for validating Romi odometry and calculating distance errors during experimentation is desired.

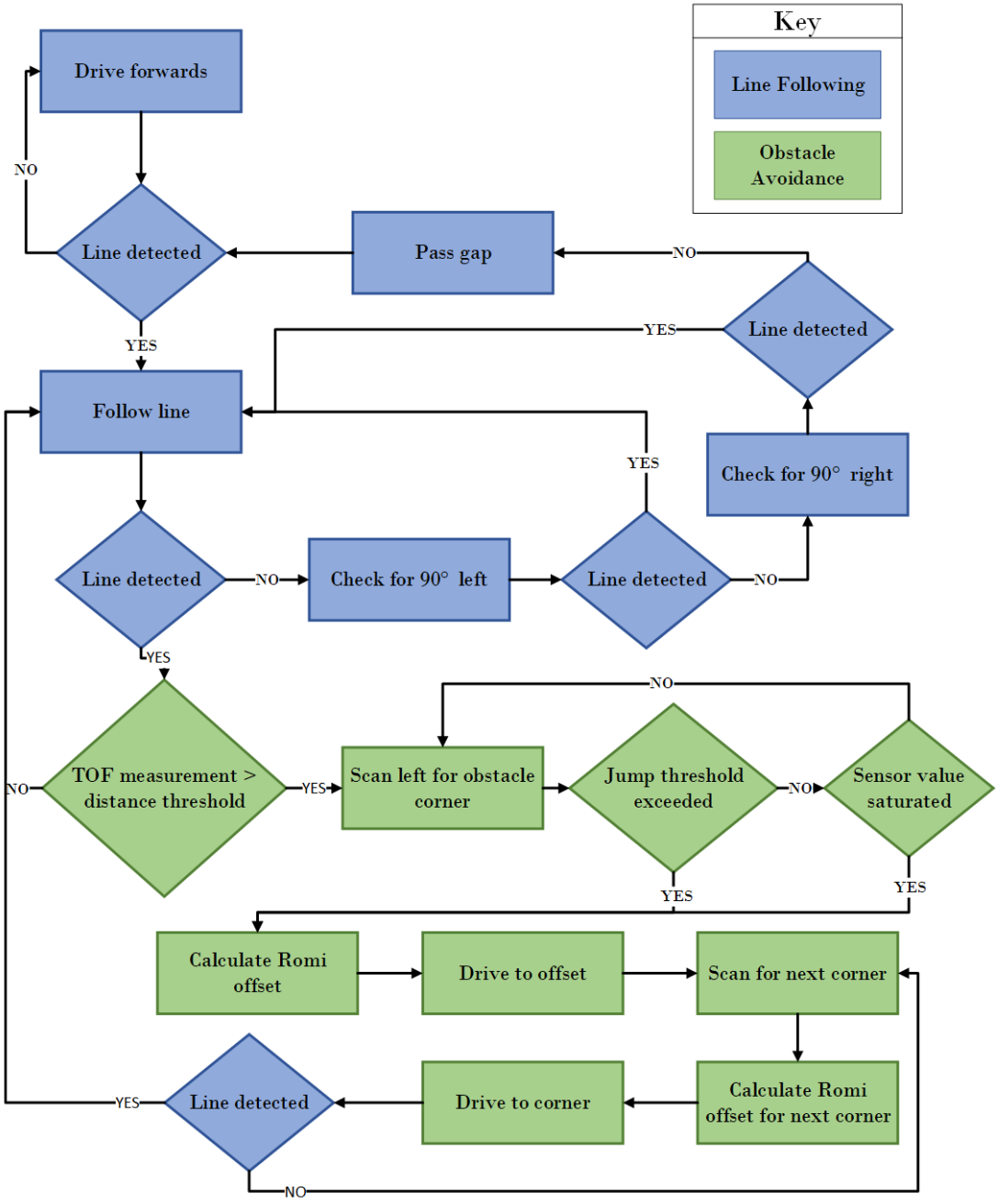
References

[1] wipro, *The rise of industrial autonomous vehicles*.
[2] Nicola Tomatis, *Obstacle Avoidance for AGVs — Dream Feature or Dangerous Gimmick?*.
[3] Pololu Robotics & Electronics, *Pololu Romi 32U4 Control Board User's Guide*.
[4] Pololu Robotics & Electronics, *QTR-MD-03A Reflectance Sensor Array: 3-Channel, 8mm Pitch, Analog Output*.
[5] Pololu Robotics & Electronics, *VL6180X Time-of-Flight Distance Sensor Carrier with Voltage Regulator*.
[6] Pololu Robotics & Electronics, *Romi Encoder Pair Kit, 12 CPR, 3.5-18V*.
[7] Dr. Paul O'Dowd, *EMATM0053: Robotic Systems*.

ACKNOWLEDGMENTS

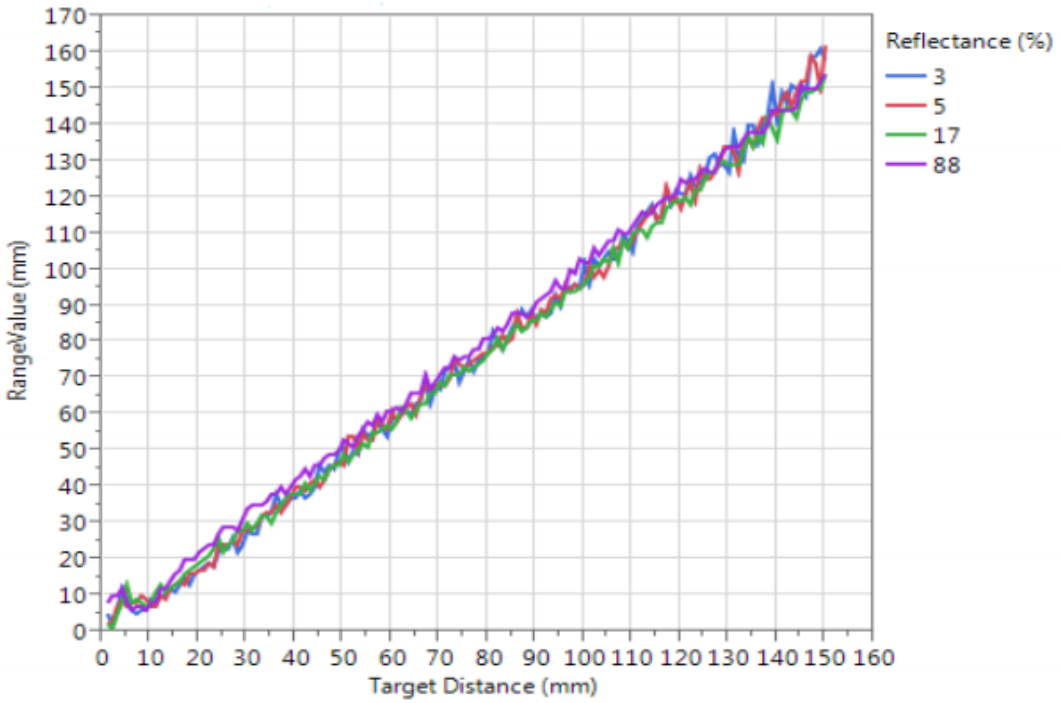
The authors would like to thank Paul O'Dowd for his organisation of the module along with the supporting labsheets. The authors would also like to thank Merihan Alhafnawi for her continued support during the weekly sessions.

Appendix A: Flowchart



Appendix B: VL6180X Datasheet

Shows the a comparison between the measured distance using the VL6180X and the actual distance. Graph taken from [5].

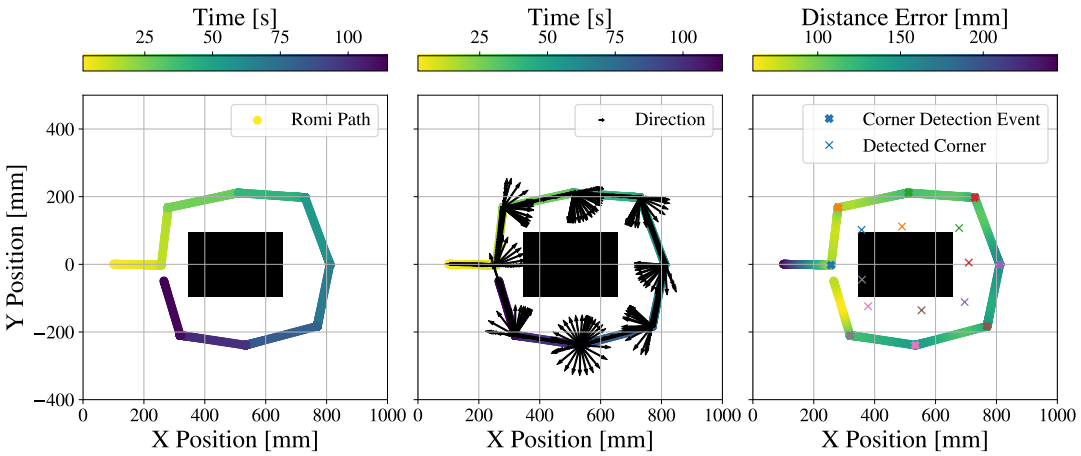


Appendix C: TOF Sensor Results

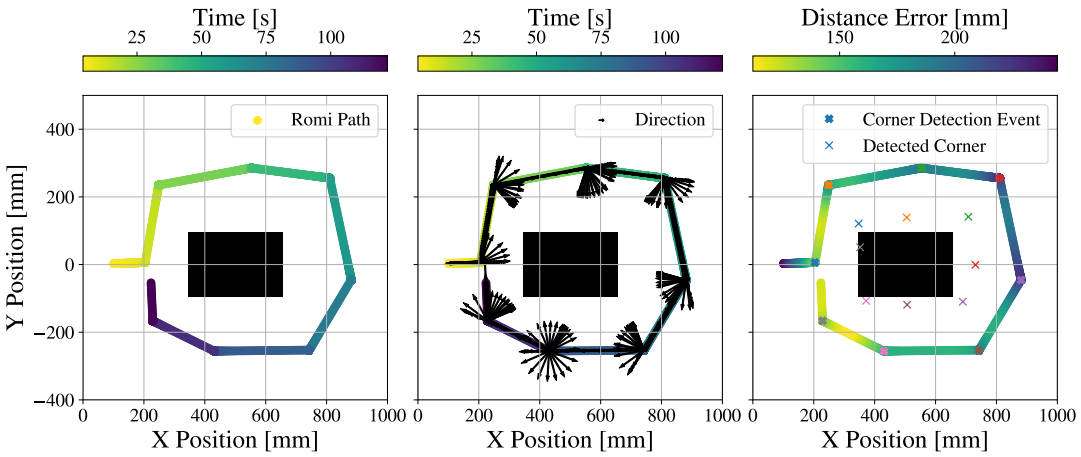
Scaling Factor	Actual Distance	Average Distance	Max Distance	Minimum Distance
1	50	49.98	52	47
	100	98.95	102	96
	150	150.48	154	147
	200	255	255	255
2	50	48.92	52	46
	100	98.46	102	96
	150	148.36	152	144
	200	197.1	202	192
	250	242.74	248	238
	300	284.86	290	278
	350	327.02	340	316
	400	359.86	370	348
	450	510.0	510	398
3	50	49.56	54	48
	100	99.15	102	96
	150	148.02	153	144
	200	193.93	198	189
	250	240.75	246	234
	300	285.52	294	276
	350	324.63	333	315
	400	362.97	372	351
	450	409.14	423	390
	500	449.39	471	429
	550	765	765	765

Appendix D: Varying Offset Distance Plots

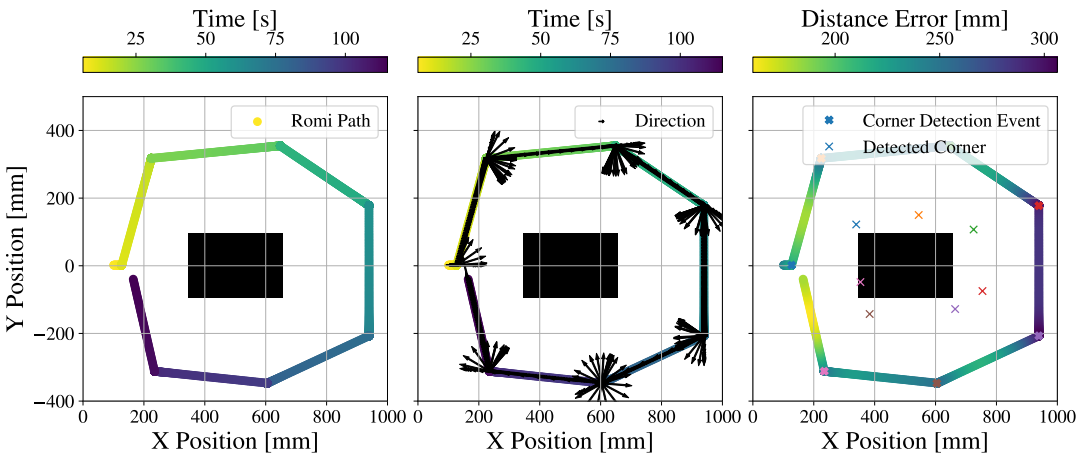
Offset distance: 100 mm, Jump Threshold: 20 mm



Offset distance: 150 mm, Jump Threshold: 20 mm

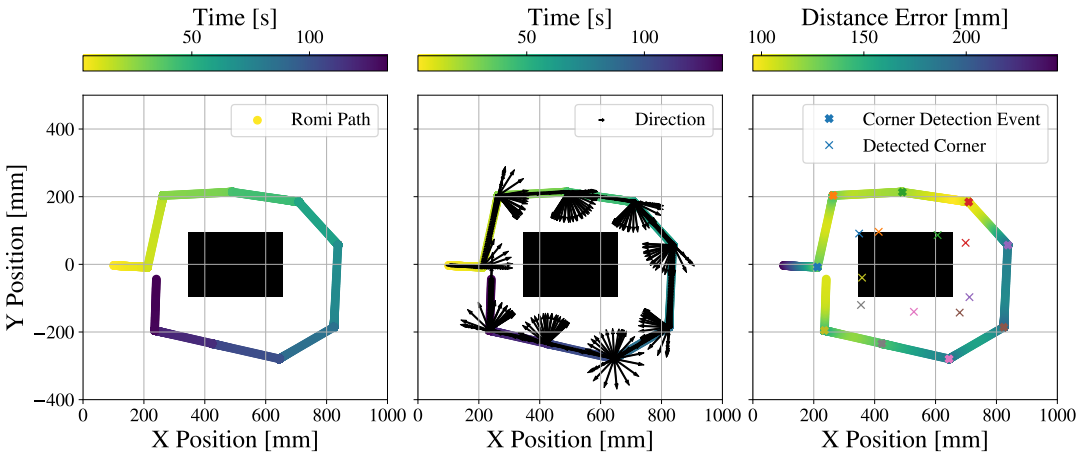


Offset distance: 220 mm, Jump Threshold: 20 mm

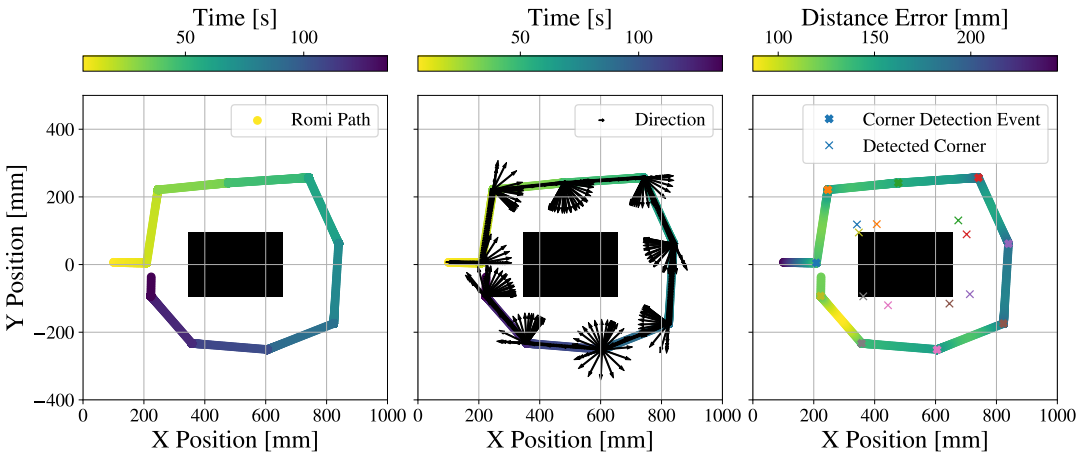


Appendix E: Varying Jump Threshold Plots

Offset distance: 140 mm, Jump Threshold: 8 mm



Offset distance: 140 mm, Jump Threshold: 10 mm



Offset distance: 140 mm, Jump Threshold: 40 mm