

Path Planning, Motion Control and Obstacle Detection of Indoor Mobile Robot

Ali Khaleel Mahmood^{a*}, Robert Bicker^b

^aHay al-jamiaa, Baghdad 14001, Iraq

^bNewcastle University, Address, Newcastle Upon Tyne NE1 7RU, UK

^aEmail: ali_khalil.mahmud@yahoo.com

^bEmail: robert.bicker@newcastle.com

Abstract

In this paper, A* path planning algorithm has been represented for a mobile robot to be able to follow a constructed path from its current position to a specified goal within its environment. To ensure that the mobile robot follow the constructed path by path planning algorithm, a motion control algorithm has been built. In the same time, to detect static obstacles and avoid collision with them, an obstacle detection algorithm has been used as a final algorithm that will be used as a part of the whole system to give the robot the ability to move from its initial known position to a specific goal in an optimum way.

Keywords: Kinematic; A* Algorithm; Odometry; Kinematic.

1. Introduction

Mobile robotics is a growing part in the robotics field, which is becoming progressively more important, useful, and commonplace in modern society. For the reason that they are providing the functionality, mobility, and multi-media possibilities in all areas of everyday life: at home, at work, in public environments, and in remote locations such as space and deep sea, they are being employed not only in industry but also in service-oriented applications. The ability of mobile robots to perform a significantly extended range of tasks and services many locations offer a great request for specialized applications that lead to increase the interest in mobile robots. This leads to increase the interest in mobile robots [1].

* Corresponding author.

The mobile robots have potential application in many areas for example:

- Service Robots:
 - Transportation of food and medication.
 - Automatic cleaning of large areas (e.g. airports, industrial sites, supermarkets, domestic vacuum-cleaner).
 - Client support (e.g. exhibitions guides, Museum tours)
 - Agricultural: Planting, fruit and vegetable picking.
 - Forests : Cleaning, tree cutting and fire preventing
- Space:
 - Space exploration
 - Remote inspection of space stations
 - Elderly and Handicapped
 - Assistance to handicapped or elderly people, helping in transportation, healthcare [2].

The aim of this study is to design, analyze, assess a set of algorithms to give the robot the ability to follow the optimum path when move in known environment from known position to a specific goal.

To achieve the above aim, three main algorithms will be built. Firstly, a path planning algorithm to construct the required path. Secondly, a motion control algorithm to successfully follow the constructed path from the first algorithm. Finally, An obstacle detection algorithm to ensure that the robot will not hit any object while move to its goal.

2. Robot Kinematic

To be able to create a control program for mobile robot hardware, it is important to understand the mechanical behavior of that robot. The process of understanding the behavior of a mobile robot starts with the procedure of describing each wheel contribution for the robot motion hence; each wheel has a role in final robot movement.

This section describes the construction of simple forward kinematic model of the robot motion. Then derives equations for combine the kinematics construction of wheels then combine them with the kinematic constraints of the robot to express the whole robot's kinematic. With these equations, the robot's maneuverability will be defined and the path of the robot will be evaluated.

The NI LabVIEW Robotics Starter Kit robot has two wheels, each with radius r . Given a point P in the center between the two drive wheels; L is the distance from the wheel to P, the spinning speed of the right and left wheels are $\dot{\psi}_R$ and $\dot{\psi}_L$ respectively; θ represent the angle.

The overall speed of the robot will be predicted by the forward kinematic in the robot reference frame:

$$\dot{\xi}_R = \begin{bmatrix} \dot{X}_R \\ \dot{Y}_R \\ \dot{\theta}_R \end{bmatrix} = f(L, r, \theta, \dot{\psi}_1, \dot{\psi}_2) \quad (1)$$

By supposing that the local reference frame of the robot is aligned such that the +XR of the robot is in the direction of movement as shown in Figure (2)

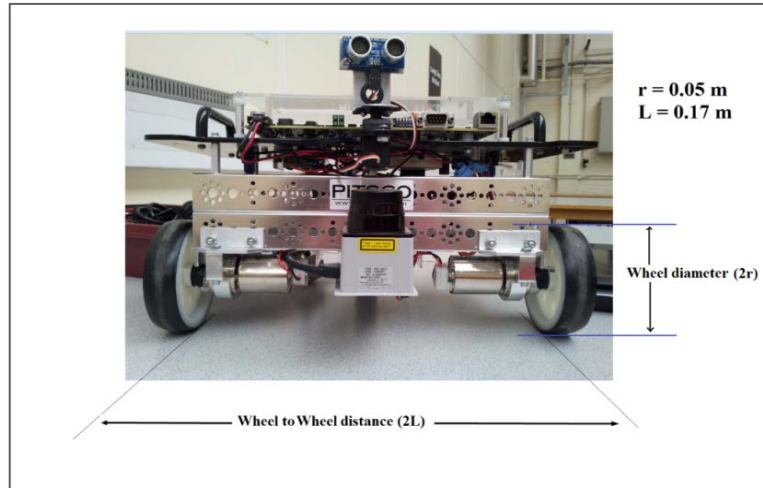


Figure 1: Wheel to wheel distance and wheels radius

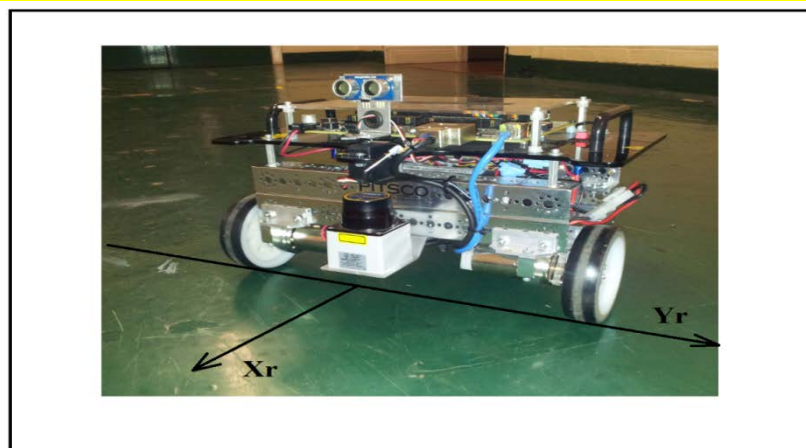
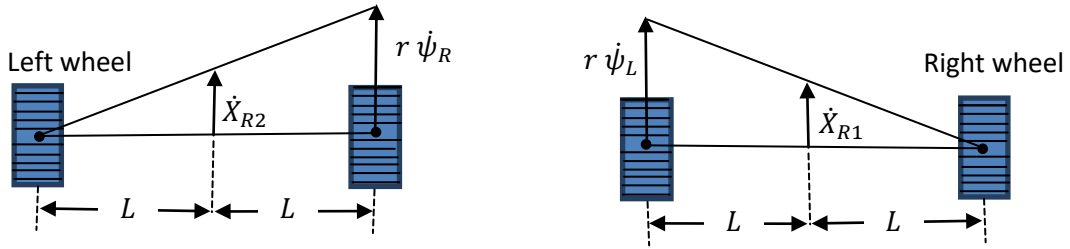


Figure 2: Mobile robot coordinate

If one wheel spins while the other wheel is stationary and contributes nothing, because point P is halfway between the two wheels, the robot will move instantaneously with half the speed:

$$\frac{r \dot{\psi}_L}{2L} = \frac{\dot{X}_{R1}}{L} \Rightarrow \dot{X}_{R1} = \frac{1}{2} r \dot{\psi}_L \quad (2)$$



$$\frac{r \dot{\psi}_R}{2L} = \frac{\dot{X}_{R2}}{L} \Rightarrow \dot{X}_{R2} = \frac{1}{2} r \dot{\psi}_R \quad (3)$$

In a differential drive mobile robot, these two contributions can simply be added to calculate the \dot{X}_R of $\dot{\xi}_R$

$$\dot{X}_R = \frac{1}{2} r \dot{\psi}_L + \frac{1}{2} r \dot{\psi}_R \quad (4)$$

The value of \dot{Y}_R is always zero because the wheels cannot contribute to sideways motion in the reference frame of the robot $\dot{Y}_R = 0$. Finally, it is essential to compute the rotational component $\dot{\theta}$ of $\dot{\xi}_R$. The contributions of each wheel can be computed independently and just added

$$w1 = \frac{-r \dot{\psi}_L}{2L} \quad (5)$$

$$w = \dot{\theta} = \frac{-r \dot{\psi}_L}{2L} + \frac{r \dot{\psi}_R}{2L} \quad (6)$$

Combining these individual formulas ($\dot{X}_R, \dot{Y}_R, \dot{\theta}_R$) yields a kinematic model for the differential-drive mobile robot:

$$(7)$$

$$\dot{\xi}_R = \begin{bmatrix} \frac{r \dot{\psi}_L}{2} + \frac{r \dot{\psi}_R}{2} \\ 0 \\ \frac{-r \dot{\psi}_L}{2L} + \frac{r \dot{\psi}_R}{2L} \end{bmatrix}$$

3. Path-Planning, Motion control and Obstacle Detection Subsystems

Find-path problem or Path planning is popular in robotics since it has an essential role in the navigation of autonomous mobile robots. Accurate path planning is very important for the robot to be able to track an optimal path between start position and goal position without collision with obstacles. According to [3], there are two main types of path planning: global path planning and local path planning. Global path planning needs all terrain to be static. In addition, there should be a complete awareness about the environment. In contrast, the implementation of the local path planning is while the robot is moving. Therefore, the local path planning has the ability to produce a new path when there is a change in the environment. For this project the local path planning will be used since the robot needs to check if there is an obstacle within its path to avoid it by change its path [3].

The general idea of local path planning is that, based on the current knowledge, a path is selected between two points. Then, when the mobile robot gets new information from the sensor about the environment, the previous path will be updated since there could be an obstacle in the selected path between starting point and the goal point [4].

For this project, the following steps will summarize the use of path planning and obstacle detection:

- Step 1: Path planning: Based on the current knowledge of the environment, a path between the current node to the target node will be planned using A* path planning algorithm.
- Step 2: Sensing: The mobile robot will take the sensor data and detect the obstacles position. If there is no obstacle within the planned path go to step 4, else go to step 3.
- Step 3: Re-plan the path: For the reason that the previous path is not available hence there is an obstacle across this path, the A* path planning algorithm will be used again for path planning.
- Step 4: motion control: After detecting the final path of the mobile robot, the motors velocity should be adjusted in order to follow the desired path [2].

3.1 The Basic Approach of A* Path planning Algorithm

A* algorithm is used to select the shortest path between start and goal nodes by efficiently compute optimal solutions when combine features of two searches; the uniform-cost search and pure heuristic search. For the A* search algorithm, the cost associated with a node is $f(n)$ which estimates the lowest total cost of all paths going through node n.

$$f(n) = g(n) + h(n)$$

Where:

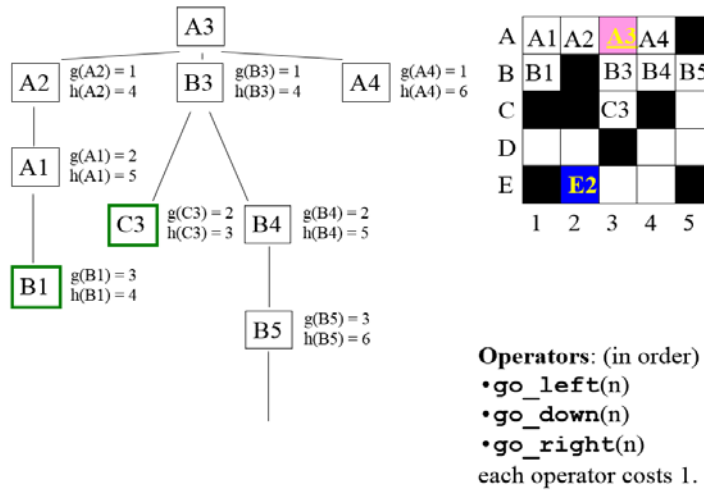
$g(n)$ = the cost of the path from the initial state to node n.

$h(n)$ = the heuristic estimate or the cost of a path from node n to a goal.

After finding $f(n)$ for each point, the node with the lowest $f(n)$ will be chosen for the expansion [5].

Terminology:

- 1- Calculate $g(n)$ the cost (or distance) of the path from the starting point to any vertex n .
- 2- Calculate $h(n)$ (the heuristic estimated cost from any vertex n to the goal).
- 3- Calculate $f(n) = g(n) + h(n)$. The value of $f(n)$ will be used by A^* to select the next step in the path.



Problem: To get from square **A3** to square **E2**, one step at a time, avoiding obstacles (black squares).

Figure 3: A* path planning steps [6]

It is important to notice this approach has the following assumption: Firstly, the region of interest is larger than the sensing range of the sensor. Secondly, the sensor has the ability to measure 180° (from -90° to $+90^\circ$). Finally, the initial position of the robot is known.

3.2 Motion control

Odometry is the most commonly used method for the instantaneously position determining of the mobile robot, hence real time positioning information is easily accessible for positioning information [7].

Odometry can be defined as the estimation of the change in the position of a mobile robot over time by using the data from motion sensors (i.e. optical encoder). In wheeled mobile robot, the odometry used to estimate its position relative to a starting location. The main benefit of using odometry in most practical applications is that it provides easily accessible to positioning information for real time position measurements.

However, this method is very sensitive to errors that produced as a result of integration the velocity measurements over time to give the robot position estimates. For this reason, this method needs calibration and processing to give accurate data collection [8]. After calibrating the encoder to give the nearest readings to the real robot movement, it will be a feedback signal for the motion control.

After achieving the path planning algorithm, it is essential to give accurate values for the motor velocities to ensure that the robot follow the required path. Because the map of the environment is divided into cells, the robot movement can be either diagonal or orthogonal.

As shown in figure (4), there are eight possible directions to move (A, B, C, D, E, F, G and H)

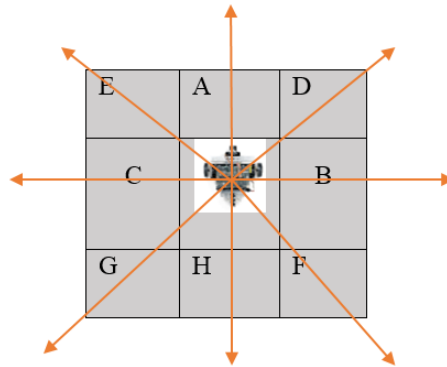


Figure 4: Possible directions of robot movement

To simplify the algorithm, all these cases can be achieved by using the same forward and angular velocities just by adjusting the time of movement. The following flowchart describes the complete process of motion control:

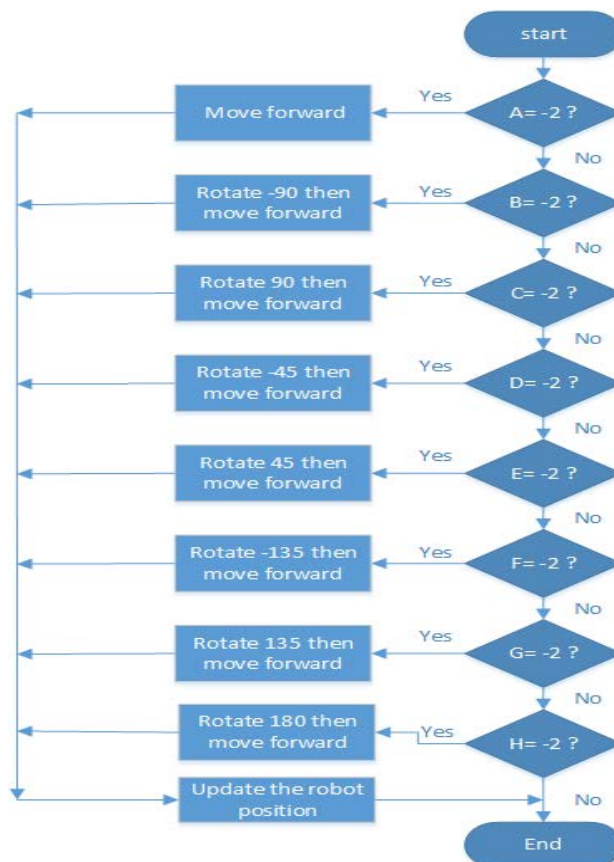


Figure 19: Motion control algorithm

3.3 Obstacle Detection Subsystem

Obstacle detection algorithm was integrated with the path planning algorithm hence it has been used to modify the direction of the mobile robot based on obstacles detection in its path. It executes this algorithm to search for the obstacles, if any obstacle is found on the mobile robot path, the old path will be replaced with a new one to avoid that obstacle as shown if Figure (5) and Figure (6) where Figure (5) shows the original path before executing the obstacle detection algorithm while Figure (6) shows the modified path depending on the sensor readings. In Figure (6) the blue squares represent the obstacles that are not included in the map of the environment. For this reason it is important to integrate the path planning algorithm with the obstacle detection algorithm.


1	1	1	1	1	1	1	1	100
1	1		1	1	1	1	1	100
1	1	-2	1	1	1	1	1	100
1	1	-2	1	1	1	1	1	100
1	1	1	-2	1	1	1	1	100
1	1	1	1	-2	-2	-2	1	100
1	1	1	1	1	1	1	Target	100

Figure 5: planned path between the starting point and the target


1	1	1	1	1	1	1	1	100
1	1		1	1	1	1	1	100
1	1	-2	1	1	1	1	1	100
1	-2	100	100	100	1	1	1	100
1	1	-2	1	100	100	100	100	100
1	1	1	-2	-2	-2	-2	1	100
1	1	1	1	1	1	1	Target	100

Figure 6: Modified path after map observation

The sensor information will be used to continuously update the path. For this reason, the obstacle avoidance algorithm with the path planning algorithm will constantly re-plan the path so this continuous re-planning allows the mobile robot to deal with dynamic environments in addition to avoid collision.

The Scanning range finder URG-04LX will be used to measure the environment. It will be used for the obstacle avoidance in addition to using it in the localization algorithm [7].

4. Software methodology

The LabVIEW Robotics Module includes several path planning algorithms including A* [9]. The following example will be used in the main program with some modifications to be compatible with the main localization algorithm. The following steps have been followed:

- **Building the Map :**

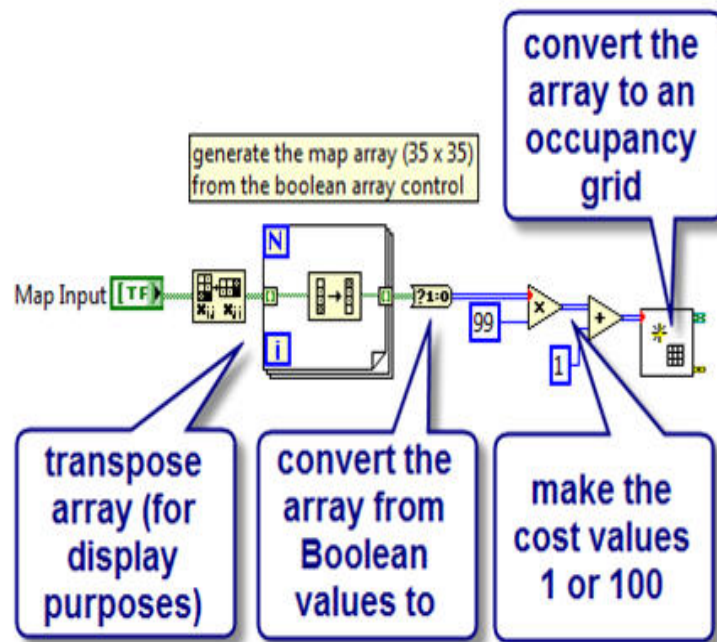


Figure 20

In LabVIEW Robotics, a 2D array of non-zero values is used to represent a map where each value in this map is represented by a cost that associated with moving from one node to another. For this project, a simple map of two values will be used, therefore, a 2D array of Boolean values will be used as a map input.

- **Plan the path:**

The next step after building the map is plan the path from one position to another. To plan a path, it is required to detect the start and target positions on the map to be able to apply A* algorithm in order to provide the best

path between them.

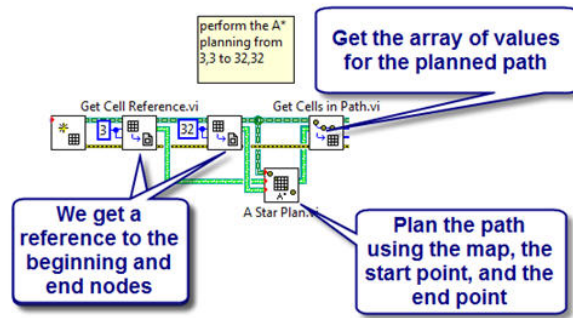


Figure 8: path planning portion of the LabVIEW code [7]

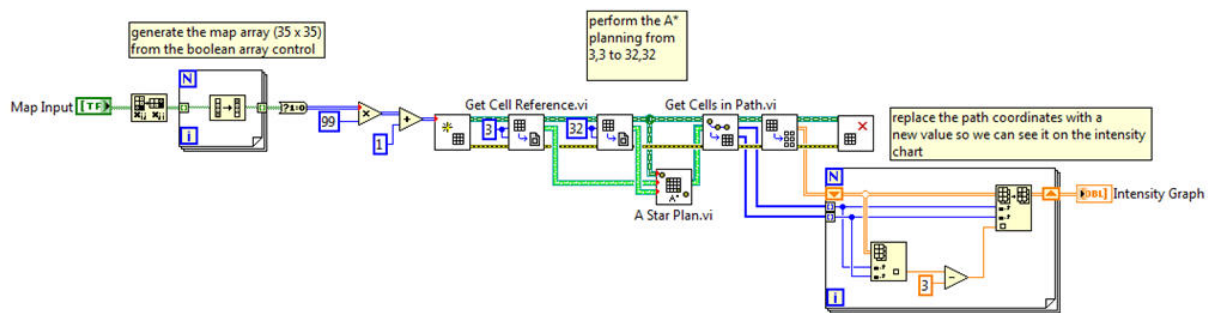


Figure 9: Final complete algorithm [9]

- **Display the complete algorithm, the maze and the planned path.**

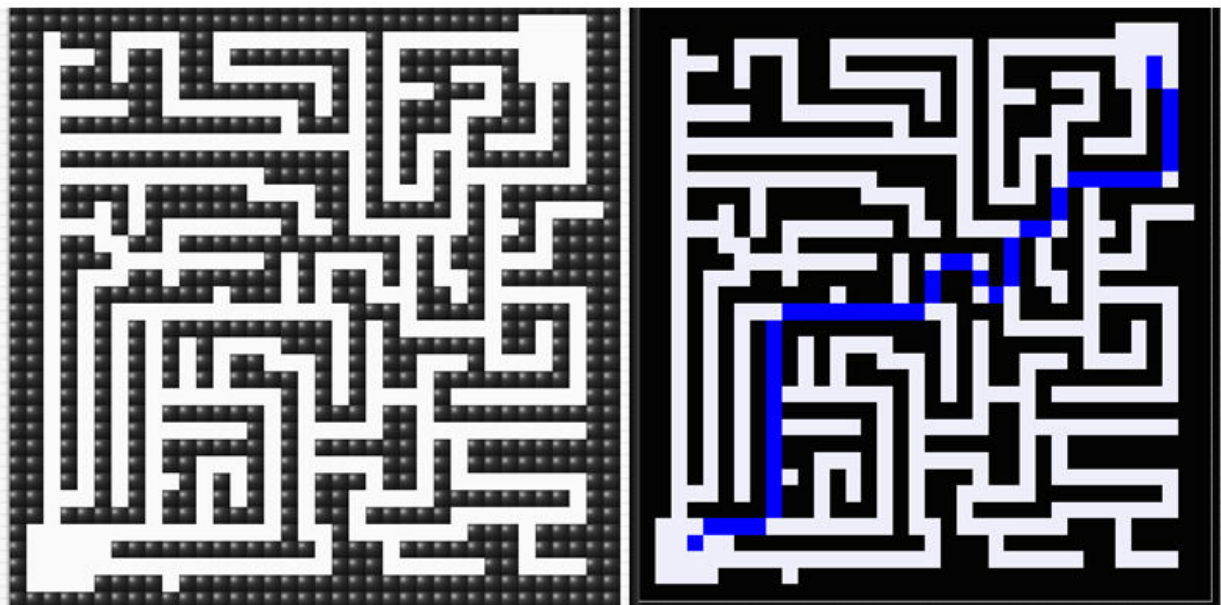


Figure 10: End result of the A* path planning algorithm

The end result is shown in Figure (10) :

It is best to understand the algorithm conceptually so that you can use it most effectively. In this tutorial, you will learn the concepts behind A* as well as see a simple example that implements A* using NI LabVIEW.

The final path planning and obstacle detection:

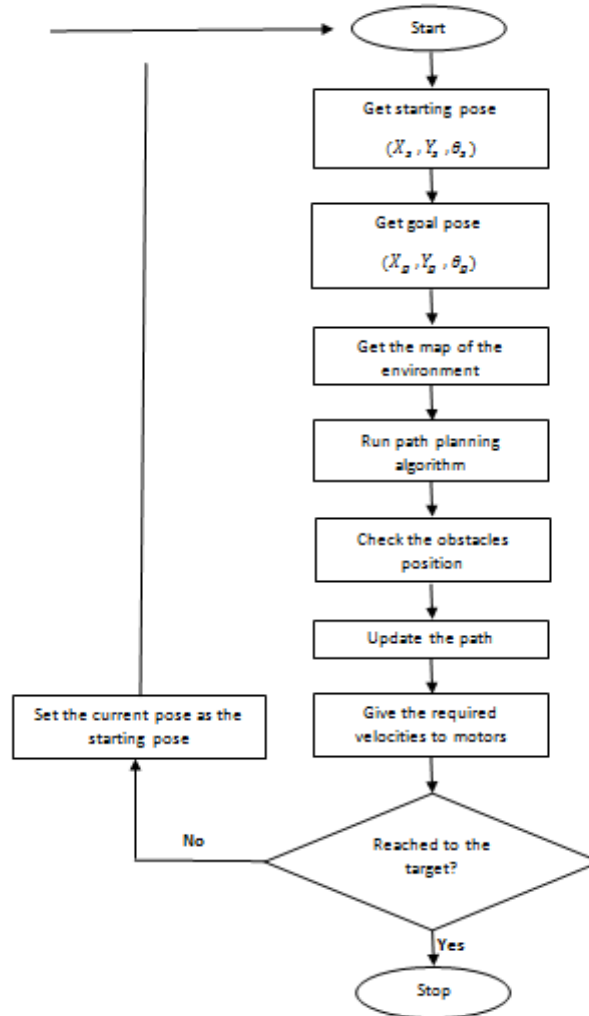


Figure 21

5. Test and Calibration

This section includes the calibration of the system component to be ready and efficient to use in the program. This include two main parts, the first part include the odometry calibration while the second part is the calibration of the sensor readings.

5.1 Odometry Calibration

For odometry calibration, two main were used; the first test is for the linear odometry error and the second test is for the angular error

- **Linear odometry calibration**

For the linear calibration, known velocities was given to the motors; for a specific amount of time (t), the robot will move a distance (d). The error in the readings can be found by comparing the distance calculated theoretically, by applying robot kinematics, with the value measured practically.

From equation (3-4), $\dot{X}_R = 0.025(\dot{\Psi}_R + \dot{\Psi}_L)$

Because $\dot{X}_R = \frac{d}{t}$, $t = \frac{d}{\dot{X}_R}$

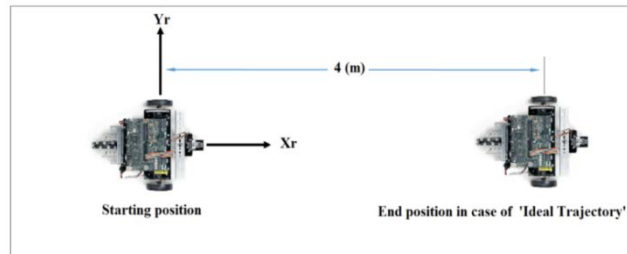


Figure 11: Linear odometry test

This equation will be used to calculate the required time of the robot movement to achieve the required distance (d). There are three different set of velocities have been used to test the linear encoder error; the distance used for this test is (4m) so the X =4m (400 cm) and Y =0 (forward displacements only)

Test1: Motors velocity ($\dot{\Psi}_R = \dot{\Psi}_L = 5 \text{ rad/s}$), d=4m

Calculation of t value:

Test 2: Motors velocity ($\dot{\Psi}_R = \dot{\Psi}_L = 10 \text{ rad/s}$), d=4m

$$\dot{X}_R = 0.025 (5 + 5) = 0.25 \frac{m}{s}, \quad t = \frac{d}{\dot{X}_R} = \frac{4}{0.25} = 16 \text{ sec}$$

$$\dot{X}_R = 0.025 (10 + 10) = 0.5 \frac{m}{s}$$

$$t = \frac{d}{\dot{X}_R} = \frac{4}{0.5} = 8 \text{ sec}$$

Calculation of t value:

Test 3: Motors velocity ($\dot{\Psi}_R = \dot{\Psi}_L = 13.33 \text{ rad/s}$), d=4m

$$\dot{X}_R = 0.025 (13.33 + 13.33) = 0.667 \frac{m}{s}, \quad t = \frac{d}{\dot{X}_R} = \frac{4}{0.667} = 6 \text{ sec}$$

Calculation of t value:

After giving motors the required velocities for the calculated period, a practical reading for the distance values have been taken. Table (1) shows the measured distance values and the error (E) for these three tests.

Error E = [calculated position – measured position]

Table 1: Comparison of actual and measured distances by encoder

Test 1		Test 2		Test 3	
X =398 cm	Y = 0 cm	X =394 cm	Y = -3 cm	X =390 cm	Y = -6 cm
E=2cm	E=0cm	E = 6 cm	E= 3 cm	E = 10 cm	E = 10 cm
X =396 cm	Y = -2 cm	X =396 cm	Y = -1 cm	X =395 cm	Y = -2 cm
E=4cm	E= 2 cm	E = 4 cm	E = 1 cm	E = 5 cm	E = 2 cm
X =396 cm	Y = -3 cm	X =395 cm	Y = -2 cm	X =393 cm	Y = -4 cm
E=4 cm	E = 3 cm	E = 5 cm	E = 2 cm	E = 7 cm	E = 4 cm
X =397 cm	Y = -1 cm	X =396 cm	Y = -2 cm	X =398 cm	Y = -2 cm
E = 3 cm	E = +1 cm	E = 4 cm	E = 2 cm	E = 2 cm	E = 2 cm

Table (1) shows that higher motors velocity leads to higher odometry error in both axes. This calibrated error will be considered in the calculation of the motion control.

- **Angular odometry calibration**

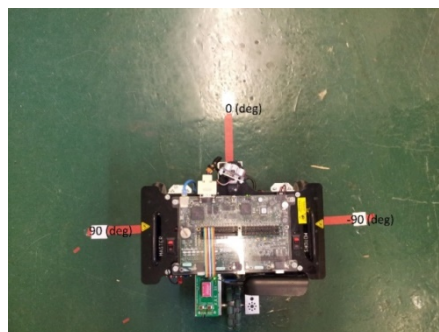


Figure 12: Starting position of the robot

In order to measure the angular odometry error, the robot has been rotated around its centre without any linear motion. For this test, motors speed are known ($\dot{\psi}_R = 5 \text{ rad/s}$, $\dot{\psi}_L = -5$); also, the angle that the robot has to rotate is known also ($\theta = 90^\circ$) and it is required to calculate the time that should be given to the robot to rotate 90° . From equation (7):

$$\dot{\theta} = \frac{-r \dot{\psi}_L}{2L} + \frac{r \dot{\psi}_R}{2L} = \frac{-(-0.05 \times 5)}{2 \times 0.17} + \frac{0.05 \times 5}{2 \times 0.17} = 1.471 \text{ rad/s}, \text{ But } \dot{\theta} = \frac{\theta}{t}, \text{ } t = \frac{\theta}{\dot{\theta}} = \frac{90 \times (\pi \div 180)}{1.471} = 1068 \text{ ms}$$

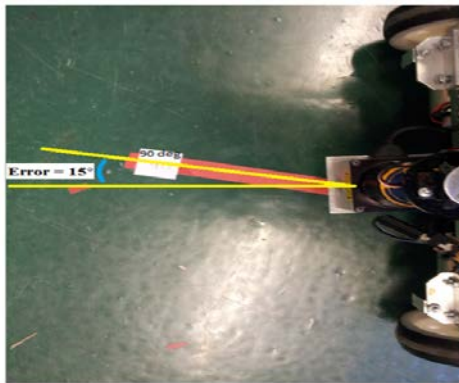


Figure 13: Required and actual robot orientations

Figure (13) shows the required and actual robot orientations, hence the robot has to rotate 90° but the actual rotation is about 105° . This test is repeated more than one time and all the results almost the same. Despite this value of error is relatively high (15° per 90°), it is easy to deal with because it is repeatable. In other words, by inserting this value in the calculations, correct rotation can be achieved.

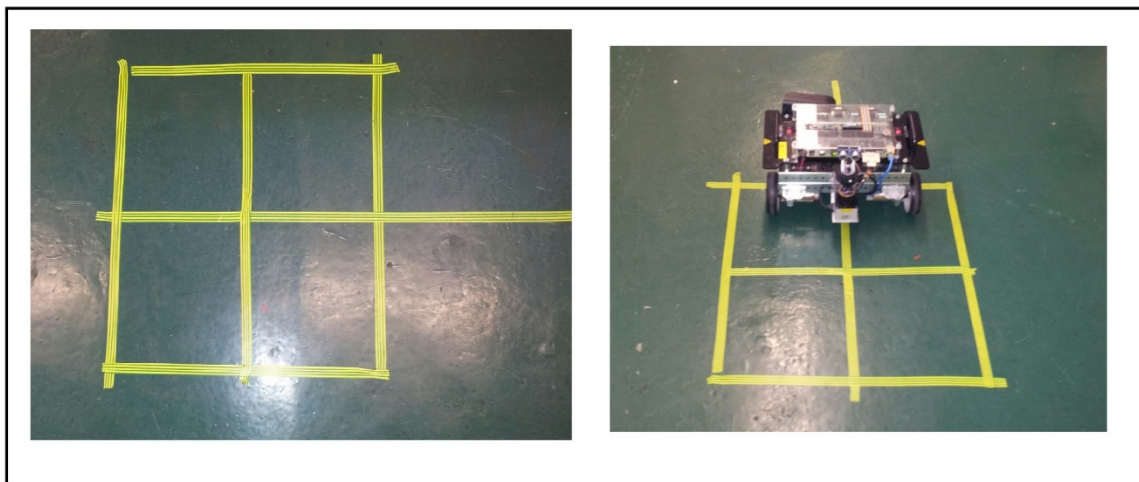


Figure 14: Final odometry test

The final step in the odometry calibration is testing the linear and angular robot displacements as shown in the Figure (14). At this test, the robot behave correctly.

5.2 Sensor Calibration

The sensor has been tested before connecting it to the robot as shown in Figure (15). This test was to find the maximum reliable range for the sensor. Despite that the sensor has the ability to measure up to 5.6m; the reliable readings are within 4m range. For this reason, the maximum range of the sensor that will be used in the program is 4m.



Figure 15: sensor calibration

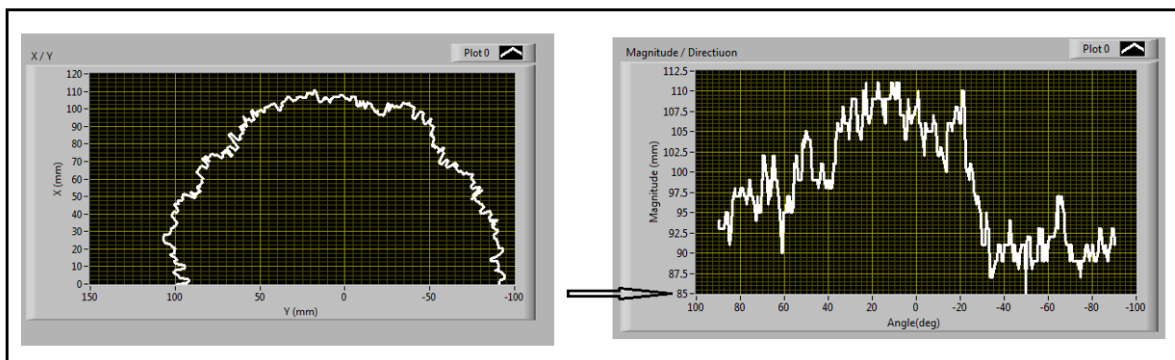


Figure 16: Sensor readings for zero distance

While doing the previous test, it has been recognized that the sensor gives zero distance reading when there is no object in its field. So, a calibration for the sensor readings has been made. The first part of this calibration is

to test the sensor readings when the actual distance from the sensor to the object is zero. As shown in Figure (16), the minimum sensor readings when there is an obstacle in the sensor field is about 85 mm. From this figure, it can be concluded that the only case that the sensor gives distance readings equal to zero is when there is no object in its field. For this reason, the sensor readings will be calibrated to give 5.6m instead of zero when the sensor field is empty.

Figure(17) shows the block diagram of the program that has been used to calibrate the sensor. The main idea in this program is comparing the sensor reading with very small value, which is 10mm in this program (or any value less than 85mm). If the sensor reading is less than 10mm, return sensor reading as 5600mm (5.6m). Else, return the same value of the sensor reading.

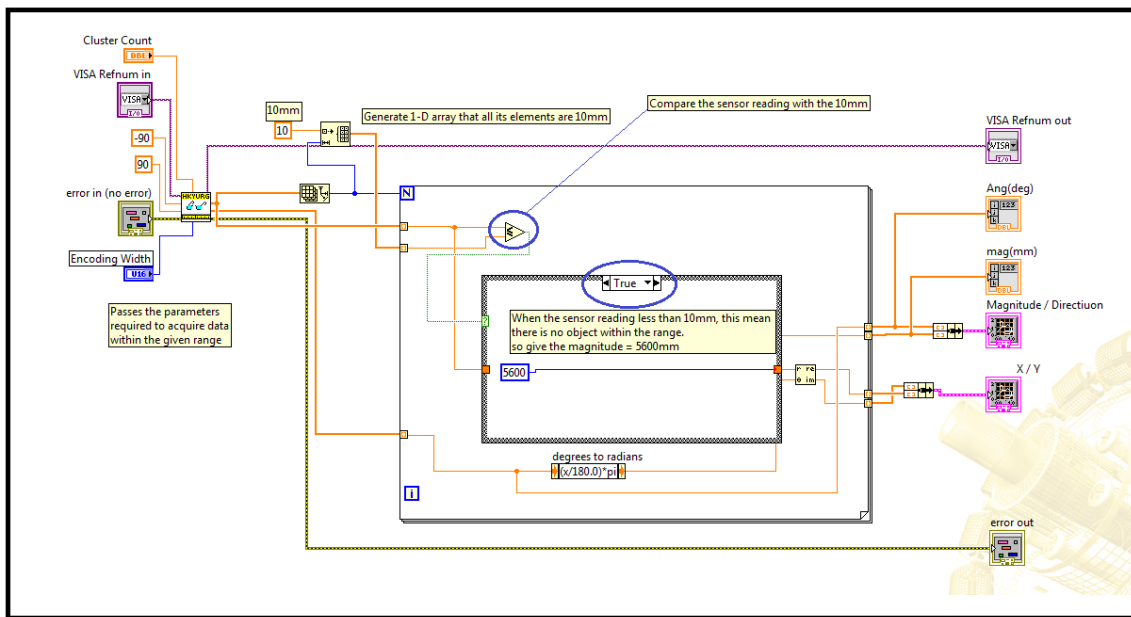


Figure 17: Block diagram of the sensor calibration program

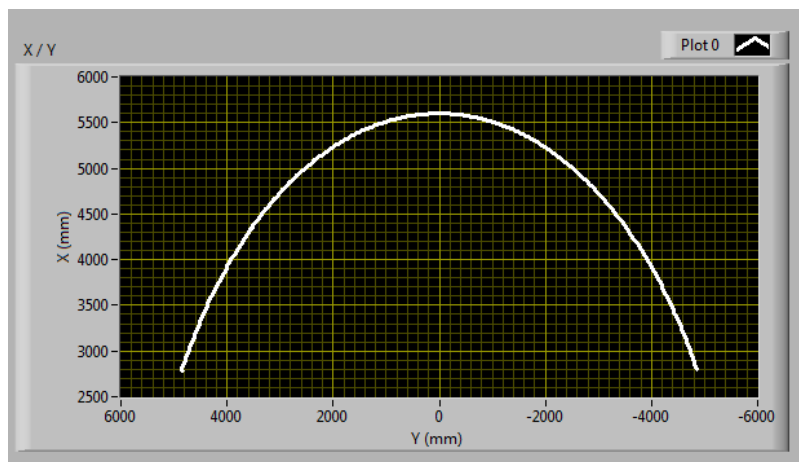


Figure 18: modified sensor readings

After achieving the previous calibration, the sensor readings became as shown in Figure(18)

6. Conclusion

The aim of the research was analyzing and implementing algorithms for a mobile robot to have the ability to follow an optimum path when moving from its position to a specific target position when put in a known environment. To achieve this aim, three algorithms have been built, tested and evaluated, path planning, motion control and obstacle detection algorithms. In addition to building these algorithms, a set of calibration steps were done. The first calibration was done for odometry. This calibration has two main calibration types; the first test was to find the linear odometry error while the second test was to find the angular odometry error. The second odometry was for the sensor that will be used to get the environment readings.

References

- [1] B. Brumson. (2012, 20 August). New Applications for Mobile Robots. Available: [http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Unsights/New-Applications-for-Mobile-Robots/content id/3362](http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Unsights/New-Applications-for-Mobile-Robots/content%20id/3362)
- [2] P. Lima and M. I. Ribeiro, "Mobile Robotics," Institute for Systems and Robotics (ISR/IST), 2012.
- [3] R. Sharma, "DESIGN AND IMPLEMENTATION OF PATH PLANNING ALGORITHM FOR WHELLED MOBILE ROBOT IN A KNOWN DYNAMIC ENVIRONMENT," International Journal of Research in Engineering and Technology, vol. 02, 2013.
- [4] T. Ersson and X. Hu, "Path planning and navigation of mobile robots in unknown environments," in Intelligent Robots and Systems, 2001 IEEE/RSJ International Conference on, 2001, pp. 858-864.
- [5] S. Mahadevi, K. R. Shylaja, and M. E. Ravinandan, "Memory of Based A-Star Algorithm for path planning of a Mobile Robot," International Journal of Science and Research (IJSR), vol. 36, pp. 37-48, 2009
- [6] Robin. (2009) Heuristic Search.
- [7] C. Popirlan and M. Dupac, "An optimal Path Algorithm for Autonomous Searching Robots," Annals for the University of Craiova-Mathematics and Computer Science Series, Vol. 36, pp. 37-58, 2009.
- [8] T. Yap Jr "Mobile Robot Navigation with low cost sensors," UNIVERSITY OF CALIFORNIA RIVERSIDE, 2009.
- [9] pjtanz. (2010, 4 August). An Introduction to A Path Planning (Using LabVIEW). Available: <https://decibel.ni.com/content/docs/DOC-8983>.