# Implementation of Mobile Robot Localisation and Path Planning for Navigation in Known Map

S.Gunasagaran[1], K.Kamarudin[1,2], A.S.A.Yeon[2], R.Visvanathan[2], S.M.Mamduh[2] and A.Zakaria[1,2]
[1]*School of Mechatronic Engineering, Universiti Malaysia Perlis, Pauh Putra Campus, Arau, Perlis, Malaysia.*
[2]*Centre of Excellence for Advanced Sensor Technology (CEASTech), Universiti Malaysia Perlis, Arau, Perlis, Malaysia.*
*kamarulzaman@unimap.edu.my*

***Abstract*—This paper describes the implementation of a laser scanner to localise a mobile robot in a known map and navigate to a pinpointed location while avoiding any obstacles within the path. The ability to successfully localise itself is a key requirement for any mobile robot. The Turtlebot Create is used as the platform part to test the localisation. The probabilistic localisation method, the Adaptive Monte Carlo Localization (AMCL) technique is used for the purpose. The approach is experimentation driven due to unique and unanticipated challenges present in the environment. As for obstacle avoidance and path planning, the Dynamic Window Approach (DWA) and Dijkstra's algorithm is used respectively. The implementation is done using Robot Operating System (ROS) framework and thus is reusable in any future projects with both, simulated and real platforms. The experiments were run in CEASTech and UniMAP Solutions, and it was observed that the robot was able to firstly map the two environments, localise itself in the known map and manoeuvre to the pinpointed locations without any collision.**

***Index Terms*—AMCL; Laser Scanner; Robotics; ROS; SLAM.**

## I. INTRODUCTION

Localization has become one of the fundamental requirements in robot nowadays. The process enables the robot to identify its own location either in absolute form, relative to the map or at least relative to certain initial position. For map-based localisation, the robot can identify its position based on information from sensors and the map itself. There are two methods for the robot to gain access to its location which is through a map fed by the user or through simultaneous localisation and mapping (SLAM).

This project is concerned about localisation of a mobile robot in a known map. Laser range finder was used to gain the obstacle locations. The data will then be fed to AMCL algorithm to enable pose estimation. At the end of the project, the robot is expected to move to a desired instructed point on the map. Mobile robot localisation is more and more in demand in today's world. Localization is where the robot can identify its position concerning the global or reference frame. For this, all the sensors need to work in line with the robot. One of the critical aspects is obstacle detection process, which is quite challenging since the robot has to navigate and reach the desired location. This can be achieved by using a laser scanner. The most current mobile robot cannot move autonomously to its desired point, and this needs more human effort to do it through wired or wireless process. The objective of this project is; (i) to explore, study and implement the methods for Simultaneous Localization and Mapping (SLAM) to obtain UniMAP Solution's Map and improving the CEASTech's Map, (ii) to implement robot localization in a known map (CEASTech's Map) using Adaptive Monte Carlo Localization (AMCL) algorithm and (iii) to implement and integrate the localization, obstacle avoidance and path planning algorithm to allow the robot to move to the desired point on a map. This project involves simulating robot localisation through Robot Operating System (ROS) and then implementing it on the real robot.

This paper is structured as follows: Section II addresses the researches related to the problem and the techniques for mapping, localisation, obstacle avoidance as well as path planning. Section III describes the steps where the process done using Robot Operating System (ROS) as well as the components and equipment needed for the project is explained. This is the list of procedure to complete the project. Section IV explains the results obtained while Section V summarises the research.

## II. LITERATURE REVIEW

Multiple types of the robot have been used in the industrial field since early 1960's. Unimate was the first industrial robot, where it was merely comprised of an arm which was used for simple tasks such as moving hot die-casts and welding car parts [1]. Mobile robots can be used for transportation and exploration [1]. Although this can expand the use of robots to various fields, mobility itself is the root cause of a few major problems in mobile robots. This is because a mobile robot needs to localise itself as to when compared to an immobile robot. The ability to localise is needed for the robot to know its actual position in the real surrounding to make rational decisions regarding its following course of action.

Since the early of the twenty-first century, navigation has been an essential part of the interest in the field of mobile robotics. A mobile robot can move from one point to another either through manual manoeuvre or automatic movements [2]. The point refers either to an absolute location or a position relative to a particular reference. One of the positioning methods that have been used widely is the global positioning system (GPS) which can provide the coordinate on earth using satellite signals. When it comes to mobile robots, this technique is, however, unsuitable due to the weak GPS signal indoors which tends to increase the rate of error in positioning. Using the GPS concept, a few researchers found a new method known as indoor positioning which uses static WI-FI beacons installed within a building [3,4]. These beacons have a similar function as the GPS's satellites, thus providing a position to position distances to calculate coordinates. The major drawback of this method is that it is not practical for exploring new places since the beacons have to be installed beforehand. Another method that can be applied for indoor navigation is the Simultaneous Localization and Mapping (SLAM). For a robot to be able to gain an autonomous ability, it must be able to travel within the environment and accurately build a map of its surrounding while being able to localise itself within the map being built [5]. This paper goes into detail regarding two main SLAM techniques namely HectorSLAM and Gmapping as they are open source and widely used techniques for indoor navigation and mapping. HectorSLAM uses an inertial sensing system which combines both 2D SLAM and 3D SLAM based on robust scan matching and navigation technique [6]. The use of a high update rate and low distance measurement noise implemented using Light Detection and Ranging Sensors (LIDARs) estimates the robot movement in real-time. This technique, however, does not require the odometry information and hence making it possible to be implemented in mobile robots that tend to operate on the uneven ground or even aerial robots. Gmapping is another laser-based SLAM technique [7]. This is also the most widely used SLAM technique when it comes to mobile robotics. Gmapping is an algorithm proposed by Grisetti and is based on Rao-Blackwellized Particle Filter (RBPF) approach. This algorithm, however, requires a higher computational load since it needs a high number of particles to obtain a decent result. Apart from this, the depletion problem about this technique also greatly reduces the accuracy of this technique. This means that a correct hypothesis can or may be eliminated as the importance weight of particles tends to become insignificant.

Adaptive Monte Carlo Localization (AMCL) is a probabilistic localisation technique. AMCL makes use of particle filter to estimate the location of the robot in opposition to a known map [8]. Patrick Heinemann has done a major research on mobile robot localisation. Based on his studies, a few self-localisation methods that are efficient have been developed, and AMCL is the most popular method. This is because it enables a mobile robot to localise itself based on real-time condition and has an error recovery in case of errors in localisation [9].

Obstacle avoidance and path planning come into play when the robot is needed to move around autonomously either to a user-defined location or through a set of a repeated movement. Path planning is the ability of the robot to identify gaps to manoeuvre through the available map while obstacle avoidance is the ability to avoid any readily available or instantaneous obstacles found throughout the path planned. The Vector Field Histogram (VFH) was an obstacle avoidance technique developed by Borenstein [10]. This method makes use of the robot's sensors to identify objects and gaps within a map and returns an open direction for navigation that leads to the goal position. A polar histogram of a few latest sensor data is used by the VFH's probabilistic algorithm to overcome the sensor noise problem. A local occupancy grid map of the robots surrounding is created to compute the probabilities. A set of passages that can fit the robot is identified using the polar histogram. Another algorithm for obstacle avoidance is by using the dynamic window approach (DWA) by Fox, Brock and Khatib. The DWA is an algorithm takes into consideration the dynamic and kinematic limitation of a mobile robot which the VFH approach does not [11]. The basic concept is to allow the mobile robot to stop before it hits a wall or an obstacle while still considering the previously stated constraints. Following this, an optimisation is performed to find a scheme that provides the most utilisation.

The term path planning was developed for use in many fields, such as robotics, control theory and artificial intelligence (AI). Various techniques have been developed such as the A-Star, D-Star, Dijkstra as well as a Theta-Star algorithm. This paper, however, will go into detail regarding two widely used algorithms, the A-Star and Dijkstra. The Dijkstra is an algorithm developed by Edsger Dijkstra in 1959. It is a graph search algorithm used to identify the shortest path through a set of interconnected nodes. It looks on the untreated neighbours of the node nearest to the start and sets or modernises their distances (regarding cost, not some nodes) from the initial point [13]. Dijkstra's algorithm is called a single-source shortest path as it solves single-source shortest-path difficulty on a subjective, directed graph. The A-star algorithm was developed in 1968 by Nils Nilsson, Bertram Raphael and Peter Hart. The A-star algorithm is well known in the mobile robotics world as one of the best path planning algorithm that can be applied to topological or metric design space [12]. It uses a blend of both shortest path and heuristic searching to complete the path planning.

## III. METHODOLOGY

### A. Robot Operating System

As a preliminary requirement for this project, Ubuntu operating system and Robot Operating System (ROS) had to be installed. For this, the stable version of Ubuntu and ROS was recognised first. The versions chosen were Ubuntu 14.04 and the ROS compatible with it, which is ROS Indigo. ROS Indigo is a flexible open-source framework used to write robot software. This software was used as the core or central software towards completion of the project. Various packages are available in the Robot Operating System, and each package can be used to complete particular tasks. Under this circumstance, the slam_gmapping, hector_slam, Adaptive Monte Carlo Localization (AMCL), Dynamic Window Approach (DWA) and Dijkstra packages were used for completing this project.

### B. Simultaneous Localization and Mapping (SLAM)

Next, slam_gmapping and hector_slam was used to improve the map of Centre of Excellence for Advanced Sensor and Technology (CEASTech). The reason for choosing both techniques was to compare the advantages and disadvantages of using these two techniques. Several parameters had to be changed to match the mobile robot base and the laser scanner used. Both mapping processes were completed using the keyboard teleoperation function from the base station. Table 1 and Table 2 show the parameters that were changed together with their description in slam_gmapping and hector_slam technique respectively. The parameters that are not mentioned were left at its default value.

Table 1
Slam_gmapping Parameters

| Parameter Name | Default Value | New Value |
|---|---|---|
| map_update_interval | 5.0 | 2.0 |
| maxUrange | 80 | 4.8 |
| maxRange | >80 | 5 |
| minimumScore | 0 | 200 |

Based on Table 1, the default value is set higher to reduce computational load but affects the quality of map produced. The new value was chosen based on the load the controller was able to handle. Both maxUrange and maxRange parameters were set based on the Hokuyo URG-04LX laser scanner's specifications. The minimumScore parameter was increased to 200 to eliminate the mobile robot jumping issue which affected the overall mapping process.

Table 2
Hector_slam Parameters

| Parameter Name | Default Value | New Value |
|---|---|---|
| map_update_distance_thresh | 0.4 | 0.05 |
| map_update_angle_thresh | 0.9 | 0.05 |
| laser_min_dist | 0.4 | 0 |
| laser_max_dist | 30 | 4.8 |
| laser_z_min_value | -1.0 | 0 |
| laser_z_max_value | 1.0 | 0 |
| pub_map_odom_transform | true | false |

Table 2 shows the list of parameters that had to be changed for the hector_slamtechnique. Both the map_update_distance_thresh and map_update_angle_thresh parameters were reduced to increase the map update frequency which in turn makes the mapping process and visualisation smoother. The laser_min_dist and laser_max_dist were set based on the Hokuyo URG-04LX laser scanner's specifications. The laser_z_min_value and laser_z_max_value were set to zero because Hokuyo URG-04LX laser scanner does not scan vertically and also since the mapping is done in two-dimensionally. The pub_map_odom_transform parameter is set to "false" since hector_slam does not require odometry information for mapping.

Figure 1 shows the slam_gmapping process that was visualised using the ROS Visualization tool (RVIZ). Next, Figure 2 shows the hector_slam mapping process visualised using RVIZ.

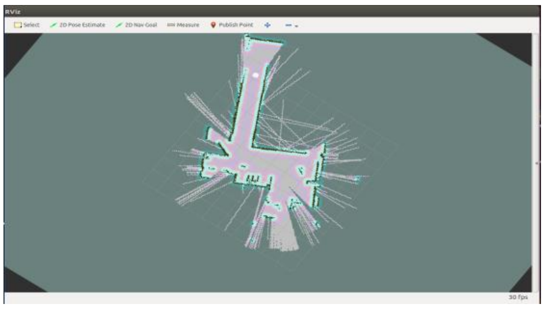
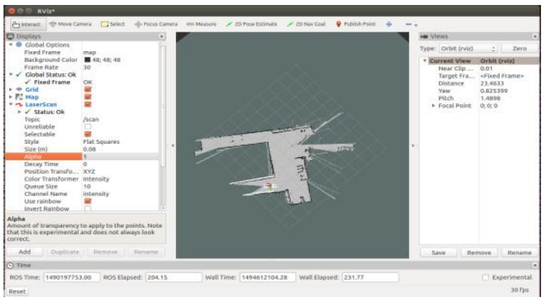


Figure 1: Slam_gmapping process



Global Options
Fixed Frame: map
Background Color: 48; 48; 48
Frame Rate: 30
Global Status: Ok
Fixed Frame: OK
Grid
Map
LaserScan
Status: Ok
Topic: /scan
Unreliable
Selectable
Style: Flat Squares
Size (m): 0.08
Alpha: 1
Decay Time: 0
Position Transfo...: XYZ
Color Transformer: Intensity
Queue Size: 10
Channel Name: intensity
Use rainbow
Invert Rainbow

Alpha
Amount of transparency to apply to the points. Note that this is experimental and does not always look correct.

Type: Orbit (rviz) Zero
Current View: Orbit (rviz)
Near Clip ...: 0.01
Target Fra...: <Fixed Frame>
Distance: 23.4633
Yaw: 0.825399
Pitch: 1.4898
Focal Point: 0; 0; 0

ROS Time: 1490197753.00 ROS Elapsed: 204.15 Wall Time: 1494612104.28 Wall Elapsed: 231.77 Experimental

Figure 2: Hector_slam process

*C. Adaptive Monte Carlo Localization (AMCL)*

Right after the mapping process was completed, the localisation technique was applied. The technique used for this was the Adaptive Monte Carlo Localization (AMCL). AMCL works by trying to match the laser scan data to the map provided and thus detecting if any drifts are occurring based on the odometry in the pose estimate. This drift is then compensated by publishing a transform between the map frame and the odometry frame such that at the end the transform map, base_frame corresponds to the real pose of the robot in the world. It is explained in Figure 3.

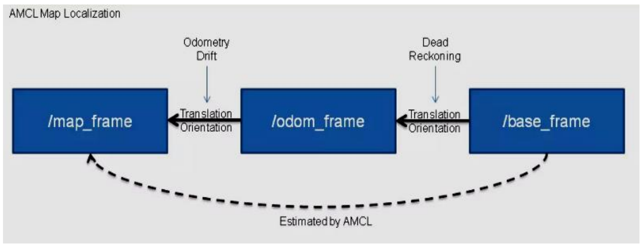


Figure 3: AMCL working principle

The improved map that was produced through the implementation of the hector_slam method on the real robot was used for this task. The AMCL technique needed many parameter adjustments before being able to localise the mobile robot. Each parameter had to be changed and carefully tested to find the most suitable for both the robot as well as the laser scanner used. Table 3 shows the parameters that were changed together with the description.

Table 3
Adaptive Monte Carlo Localization Parameters

| Parameter Name | Default Value | New Value |
|---|---|---|
| min_particles | 100 | 500 |
| max_particles | 5000 | 1500 |
| update_min_d | 0.2 | 0.05 |
| update_min_a | 30° | 10° |
| laser_min_range | -1.0 | 0 |
| laser_max_range | -1.0 | 4.8 |
| laser_likelihood_max_dist | 2.0 | 0.8 |
| odom_alpha1 | 0.2 | 0.7 |
| odom_alpha2 | 0.2 | 0.5 |
| odom_alpha3 | 0.2 | 0.8 |
| odom_alpha4 | 0.2 | 0.5 |

Based on Table 3, the min_particles and max_particles range had to be set smaller since the Hokuyo laser scanner has a higher precision as compared to the default parameter set for the Microsoft Kinect. As for update_min_d and update_min_a, these values were set lower to increase the update frequency and hence to make the localisation more precise. The laser_min_range and laser_max_range were both set based on the Hokuyo laser scanner description. The laser_likelihood_max_dist, on the other hand, was set lower to reduce the overall computational load and thus increasing the performance. Each odom_alpha value was set based on multiple tests at different places on the map. A higher value of each odom_alpha reduces the dependency of the algorithm towards the robot's odometry information.

*D. Obstacle Avoidance and Path Planning*

The next step after localisation is to integrate both obstacle avoidance and path planning algorithm into the system. First, the obstacle avoidance algorithm was added to the system.

The algorithm used was the Dynamic Window Approach (DWA).

Table 4
Dynamic Window Approach Parameters

| Parameter Name | Value |
|---|---|
| max_vel | 0.2 |
| min_vel | 0 |
| max_trans_vel | 0.5 |
| min_trans_vel | 0.1 |
| max_rot_vel | 0.2 |
| min_rot_vel | 0 |
| acc_lim | 0.05 |
| goal_tolerance | 0.05 |
| path_distance_bias | 20 |
| goal_distance_bias | 3 |
| occdist_scale | 0.01 |

Based on Table 4, the max_vel parameter is set lower at 0.2m/s than the actual robot capacity of 0.5m/s to reduce odometry error that occurs that higher speed. The min_vel is set to zero for an apparent reason so that the robot can come to a halt at any point in time. The acc_lim is set to 0.05m/s$^2$ for the same reason as setting the velocity lower which is to reduce odometry errors. The goal_tolerance, path_distance_bias, goal_distance_bias and occdist_scale was set at the lowest possible values whereby there were no errors in reaching the goal position. Any lower value causes the robot to stop when it meets an obstacle, producing an error.

The path planning, on the other hand, involved the use of Dijkstra's algorithm since it is an open source algorithm as well as the basic and easy to implement the algorithm. The Dijkstra's algorithm needs two custom parameter files to be created for it to be used. The files include the costmap parameters as well as the global planner parameters. The Table 5 and Table 6 show the list of parameters that have to be set for the costmap and global planner respectively. There was no default value for these parameters as it depends solely on each robot's design.

Table 5
Global Planner parameters

| Parameter Name | Value |
|---|---|
| use_dijkstra | True |
| use_grid_path | False |
| allow_unknown | False |
| use_quadratic | False |

Based on Table 5, the use_dijkstra parameter is set to true since that path planning algorithm was used to complete the path planning task. The use_grid_path parameter was set to false to allow diagonal movement of the robot. The allow_unknown parameter was set to false since a previously generated map is used instead of a dynamically expanding map. The use_quadratic parameter was also set to false since Dijkstra uses simple mathematics for path planning instead of quadratics method used by A-star algorithm.

Based on Table 6, the robot_radius had to be measured. This parameter is needed for path planning as it is necessary to know if the robot can pass through an obstacle or path. The obstacle_layer was enabled so that obstacle throughout the path can be detected and the costmap can be updated accordingly. As for the unknown_threshold parameter, it was based on trial and error and also the map's resolution. The obstacle_range was set lower than the maximum range of Hokuyo laser scanner since it increased the efficiency of path planning when dynamic obstacles were met. The raytrace_range was set to the default Hokuyo's configuration. The observation_sources is the list of source for detecting obstacles throughout the path of the robot. The sources were the laser scanner and also the robot's bump sensor. The data_type and topic were the type of sensor data and its respective ROS topic. The min_obstacle_height and max_obstacle_height were set to zero since the whole process was done in the two-dimensional map and both observation source does not produce value in z-domain. The inflation_radius was set to 0.2 meters to reduce computational load as well making path planning more specific to instantaneous dynamic obstacles. The global_frame was set as "map" since a static map was used rather than a dynamic continuous update map. The reason is also why the parameter static_map was set to true. The update_frequency and publish_frequency were set to 3.0Hz and 0Hz respectively to increase the rate at which obstacle is updated to the costmap for continuous and smooth path planning.

Table 6
Costmap Parameters

| Parameter Name | Value |
|---|---|
| robot_radius | 0.2 meter |
| obstacle_layer | True |
| unknown_threshold | 15 |
| obstacle_range | 2.5 |
| raytrace_range | 4.8 |
| observation_sources | scan; bump |
| data_type | LaserScan; Bumper |
| topic | scan; bumper_pointcloud |
| min_obstacle_height | 0 |
| max_obstacle_height | 0 |
| inflation_radius | 0.2 |
| global_frame | map |
| robot_base_frame | base_footprint |
| update_frequency | 3.0 |
| publish_frequency | 0.0 |
| static_map | True |

Once the path planning was completed, the AMCL, DWA and Dijkstra's algorithm has to be combined to produce a custom package for the robot to be able to localize and navigate to pinpointed location while avoiding obstacles using the catkin method.

### *E. System Architecture*

Figure 4 shows the overall system architecture. From the figure, it can be seen that the Hokuyo URG-04LX laser scanner and the robot, Turtlebot Create, communicate with the netbook using a universal serial bus (USB) connection. As for the base station, it communicates through a wireless network connection using the router. The Turtlebot Create robot was used as the robotic platform for the localisation system. It is a low-cost robot supported by open source software, the Robot Operating System (ROS). A netbook is used as an interfacing medium for laser scanner and the robot as they both need a USB connection to communicate with each other. The base station for this project is a Dell laptop which has Windows 8 operating system with Ubuntu 14.04 for Robot Operating System (ROS) installed in the VMware Workstation. A wireless router is used as the data transfer medium between the netbook and the base station. Finally, a Hokuyo URG-04LX is used for mapping, localisation as well as obstacle avoidance.

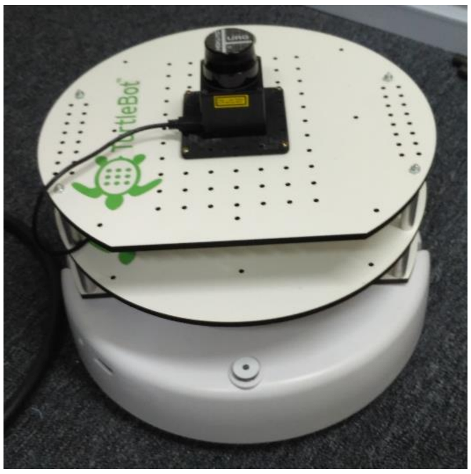


Figure 4: System architecture

## IV. Results

### *A. Mapping Process*

The mapping process was first done at the UniMAP Solution. Figure 5 shows the map generated by UniMAP Solution.

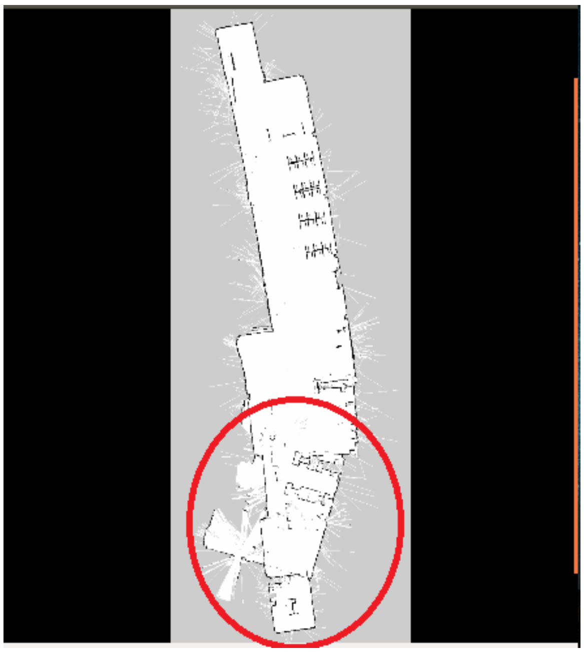


Figure 5: UniMAP Solution’s map

As it can be seen in Figure 5, the red marked is where the mapping process failed in producing a decent. This is because the environment was poor regarding features for comparison needed for map building. Another reason is that the Hokuyo URG-04LX has a maximum range of up to 4.8m and hence making it impossible to find any difference in features at a longer distance. Nevertheless, the significant portion of the map could still be used for localisation in the future. The better map could also be produced once more features are loaded in UniMAP Solution. The next task is to improve the current CEASTech’s map so that the new map covers more areas and more accurately represent the whole layout.

Figure 6 and Figure 7 show the map that was generated using slam_gmapping and hector_slam algorithm at CEASTech respectively. Decent maps were able to be generated since the environment is vibrant and there are constant changes that can be captured as the robot moves. However, the hector_slam algorithm produced a better map as compared slam_gmapping. This is because hector_slam is not affected by odometry information. It produces a map solely using the sensor data. For this cause, it can only be used in rich environments. Hence, a high update rate and the accuracy of the laser scanner played an essential role in producing the map. The marked red areas on the map generated using slam_gmapping clearly shows a minor shift of region as compared to the map produced using hector_slam.

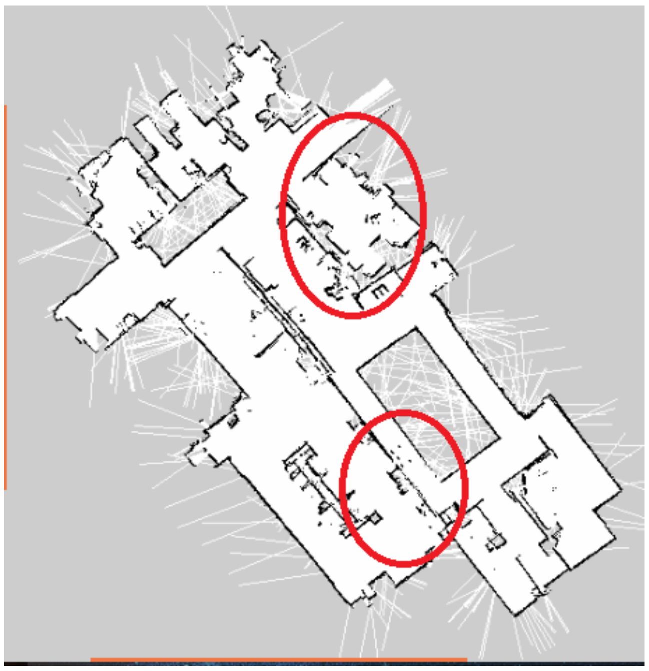


Figure 6: CEASTech’s Map using Slam_gmapping

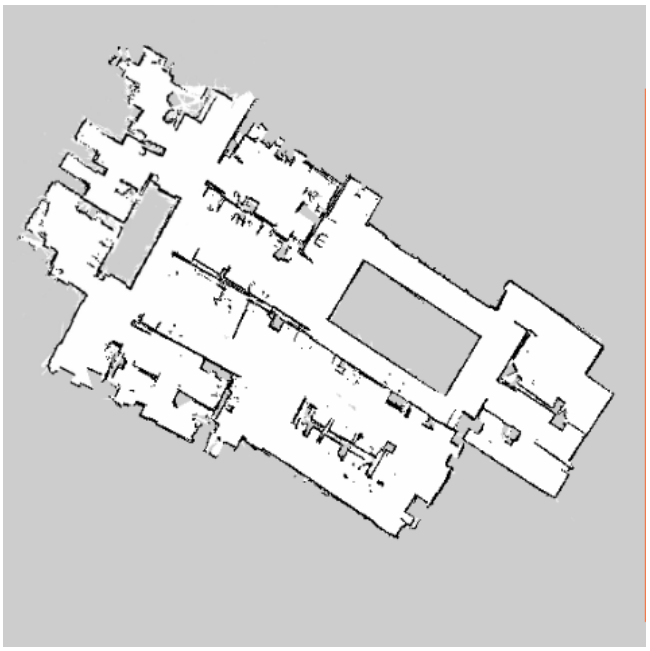


Figure 7: CEASTech’s Map using Hector_slam

Following the mapping process, the Adaptive Monte Carlo Localization (AMCL) algorithm was implemented onto the Turtlebot. The map generated using the hector_slam algorithm in CEASTech was used for this process.

Based on Figure 8 and Figure 9, the marked red area is the location of the robot before and after localisation respectively. As it can be seen in Figure 9, the robot has lined up to its actual position on the map.

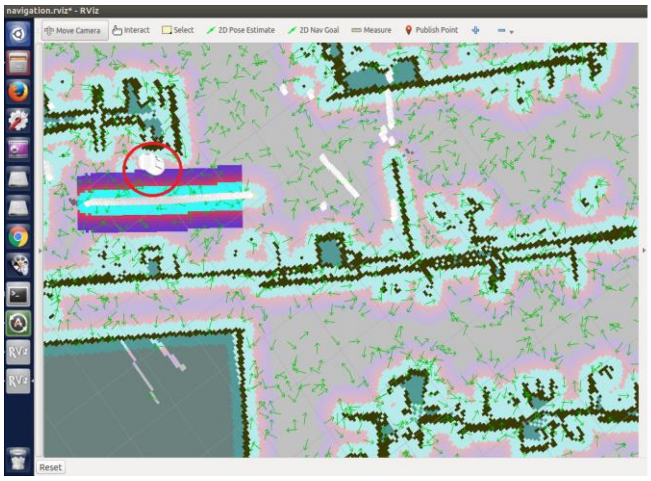


Figure 8: Robot's location before localisation

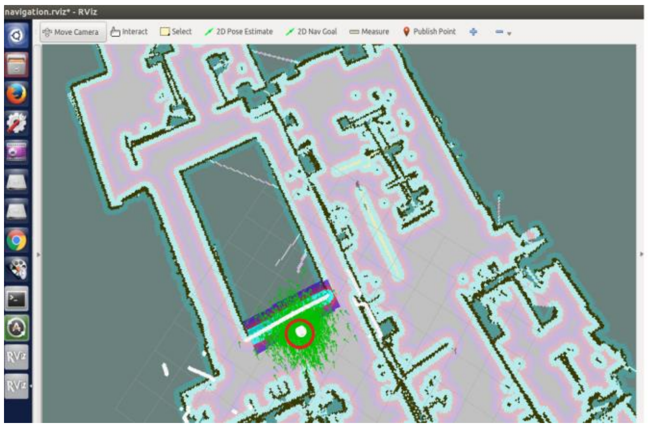


Figure 9: Robot's location after localisation

Next, the obstacle avoidance and path planning algorithm which are the Dynamic Windows Approach and Dijkstra respectively were added. This was completed by using the custom package created for the robot to localise and navigate to the goal position. Figure 10 to Figure 12 shows the result of using the custom package. Based on Figure 10, the goal position is pinpointed. The marked red area shows the robot's initial position while the black mark shows the goal position being pinpointed by the green arrow. Figure 11 shows the path generated for the robot to reach the goal position. Next, Figure 12 shows the robot has reached the goal position while successfully avoiding static and dynamic obstacles.

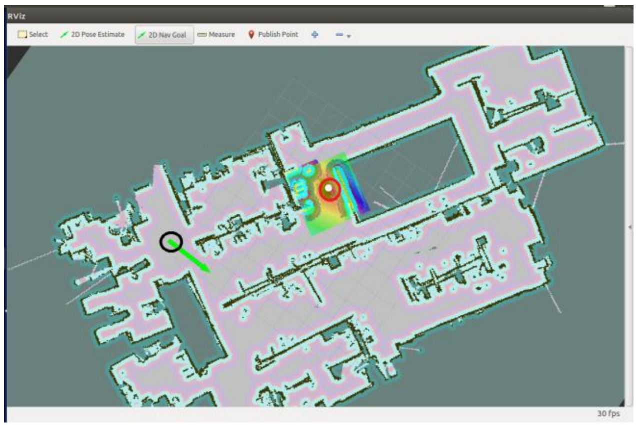


Figure 10: Pinpointing the goal position and orientation (red mark shows robot's initial position while black mark shows goal position)

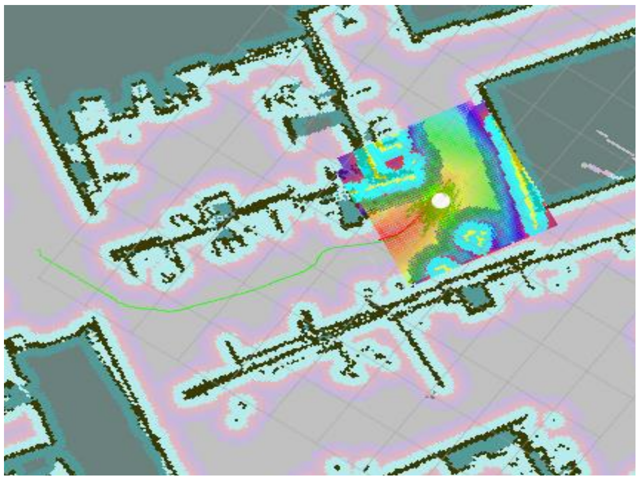


Figure 11: Path generated to reach the goal position

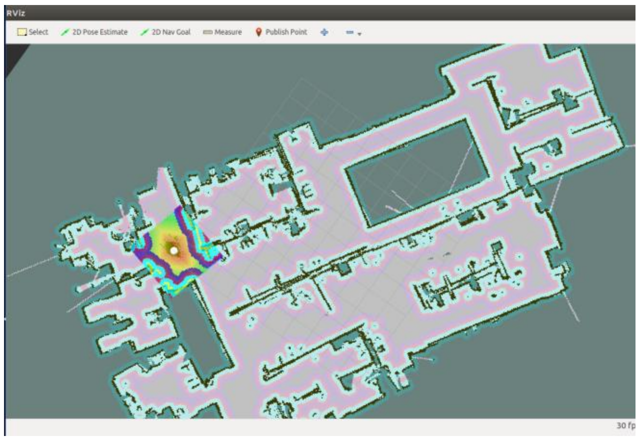


Figure 12: Robot reached the goal position

## V. Conclusion

Overall, the objective of the project has been achieved. The Hokuyo URG-04LX has been successfully interfaced to the Turtlebot Create for the use in Robot Operating System (ROS). The methods available for mapping and localisation has been extensively studied and implemented. The slam_gmapping and hector_slam were used for simultaneous localisation and mapping (SLAM) while the Adaptive Monte Carlo Localization (AMCL) was implemented for localisation. The Dijkstra's algorithm was successfully implemented for path planning while the Dynamic Window Approach (DWA) was used for obstacle avoidance. At the end of the project, the mobile robot, Turtlebot Create, can localise itself within a known map and move to any location pinpointed while avoiding any new or dynamic obstacles within the path.

## REFERENCES

[1] Robot hall of fame. 2016. [Online]. Available: http://www.robothalloffame.org/ unimate.html.

[2] Siegwart, Roland, Illah Reza Nourbakhsh, and Davide Scaramuzza. Introduction To Autonomous Mobile Robots. 1st ed. Cambridge (Mas.): MIT Press, 2011.

[3] N. Chang, R. Rashidzadeh and M. Ahmadi, "Robust indoor positioning using differential wi-fi access points", *IEEE Transactions on Consumer Electronics*, [online] vol. 56, no. 3, pp. 1860-1867, 2010. Available: ieeexplore.ieee.org

[4] S. Mazuelas, A. Bahillo, R. Lorenzo, P. Fernandez, F. Lago, E. Garcia, J. Blas and E. Abril, "Robust Indoor Positioning Provided by Real-Time RSSI Values in Unmodified WLAN Networks", *IEEE Journal of Selected Topics in Signal Processing*, [online] vol. 3, no. 5, pp. 821-831, 2009. Available: ieeexplore.ieee.org

[5] R. Havangi, "Intelligent FastSLAM: An Intelligent Factorized Solution to Simultaneous Localization and Mapping", *International Journal of Humanoid Robotics*, [online] vol. 14, no. 01, p. 1650026, 2017. Available: www.sciencedirect.com

[6] F. Koksal and C. Ersoy, "A Flexible Scalable Solution for All-Optical Multifiber Multicasting: SLAM", *Journal of Lightwave Technology*, [online] vol. 25, no. 9, pp. 2653-2666, 2007. Available: www.sciencedirect.com

[7] G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters", *IEEE Transactions on Robotics*, [online] vol. 23, no. 1, pp. 34-46, 2007. Available: ieeexplore.ieee.org

[8] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte Carlo Localization for Mobile Robots.” [online]. vol 4, Pages 243–282. Available: www.sciencedirect.com

[9] P. Heinemann, T. Rückstieß, and A. Zell, “Fast and Accurate Environment Modelling using Omnidirectional Vision,” [online] vol.1 Pages. 1–6. Available: scholar.google.com

[10] J. Latombe, *Robot motion planning*, 1st ed. Boston: Kluwer, 2010.

[11] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots", *IEEE Transactions on Robotics and Automation*, [online] vol. 7, no. 3, pp. 278-288, 1991. Available: ieeexplore.ieee.org

[12] D. Adams, "Introduction to Inertial Navigation", *Journal of Navigation*, [online] vol. 9, no. 03, pp. 249-259, 1956. Available: scholar.google.com

[13] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico and L. Jurišica, "Path Planning with Modified a Star Algorithm for a Mobile Robot", *Procedia Engineering*, [online] vol. 96, pp. 59-69, 2014. Available: www.sciencedirect.com.