

Navigation for a mobile robot to inspect aircraft

1st Thanavin Mansakul

dept. School of Aerospace, Transport
and Manufacturing
Cranfield University
Cranfield, UK
t.mansakul.621@cranfield.ac.uk

2nd Ip-shing Fan

dept. School of Aerospace, Transport
and Manufacturing
Cranfield University
Cranfield, UK
i.s.fan@cranfield.ac.uk

3rd Gilbert Tang

dept. School of Aerospace, Transport
and Manufacturing
Cranfield University
Cranfield, UK
g.tang@cranfield.ac.uk

Abstract—Preflight inspection is an important part of aircraft maintenance, as it does not affect only safety but also expenditure. In order to reduce the risks from human errors and the cost of operation, a mobile robot is introduced as an effective solution. This research examines navigation for aircraft inspection using a mobile robot. The two path planning methods, Dijkstra with DWA and A* with TEB, are compared by navigation performance in the simulation and three environments which are office, atrium, and hangar, to ensure that the robot is able to handle a variety of situations. In addition, an ultrasonic sensor was installed to perform obstacle avoidance and support LIDAR when it failed. The result is that both algorithms have their advantages depending on the purpose; if an accurate position is required, A* with TEB should be selected, while Dijkstra with DWA offers a smooth path and less time spent in small and complex environments. The notable parameters in navigation for the mobile robot, TurtleBot3, are presented, and limitations are detailed. The use of the autonomous mobile robot could support the operation with high precision and repeatability.

Keywords—Autonomous mobile robot, TurtleBot3, aircraft inspection, navigation, path planning, A*, Dijkstra, Dynamic-Window Approach (DWA), Timed Elastic Band (TEB)

I. INTRODUCTION

Aircraft require maintenance to repair and replace deteriorated parts from their flying operations. Regarding Federal Aviation Administration [1], nearly 80 per cent of aircraft inspection is a visual inspection examining surface discontinuities such as cracks and corrosion influencing structural failures. Aircraft inspection is required by regulation in aircraft maintenance to ensure safety on every flight. According to the inspection process, technical operators walk around and manually check the aircraft following the checklists [2] to investigate parts based on checklists. Imagine how many flights are each day worldwide, so it demands a number of workers working 24/7. Apparently, the procedure consumes considerable time, and any mistakes could happen due to human factor variations. Using the mobile robot could be a key solution to supporting aircraft inspections. Even though there are many elements in inspection, this research emphasises navigation, which is a significant part of integrating with vision and swarm operation in the next move.

This project aims to navigate the robot autonomously to determine waypoints. Testing on various stages is delivered to confirm that the robot can fulfil the requirements of each stage before acting on the final stage, the hangar. The robot will be tested in the simulation, office, atrium, and hangar, respectively. The objectives are measured by four indicators: (1) Task achievement, the robot will perform not only moving but also supporting aircraft inspection. If the robot is not able to move into a specific location, the inspection will not be successful because of incomplete data collection; (2) Repeatability is to ensure that the robot has the ability to do the task again and again. If the robot rarely completes the

mission, or each result is far more different despite the same mission, that is a lack of repeatability; (3) Collision-free is a safety aspect to prevent damage to lives and properties; (4) Accuracy is another crucial objective in assessing navigation performance; high accuracy definitely influences reliable and effective operation. However, if the positioning error exceeds a limitation, it will not be accepted and will be identified as a failure in achieving the task.

Challenging navigation can be divided into three parts[3]: localisation, motion control, and path planning involving obstacle avoidance. The localisation and motion control remains the same as the basic packages from TurtleBot3. However, path planning is in attention. The renowned path planning algorithms are applied for TurtleBot3 to verify performance in each environment following the benchmark. An ultrasonic sensor is also mounted in front of the robot to avoid obstacles lower than LIDAR detection. The main contribution of this work is:

- (1) A pair of well-known global and local path planning is matched and compared, namely Dijkstra with DWA and A* with TEB aiming to perform in the hangar. Also, fine-tuning notable parameters is provided;
- (2) An ultrasonic sensor is equipped, and installation is detailed to perform obstacle avoidance together with LIDAR as sensor fusion.

II. RELATED WORK

A. Navigation and obstacle avoidance

The robot is unable to locate itself in the working environment without localisation. In TurtleBot3, the distance sensor and encoder are equipped, so Adaptive Monte Carlo Localization (AMCL) can be used as the primary localisation system of the robot. About AMCL, real-time performance was improved from Monte Carlo localisation (MCL) to obtain the robot position on the map, such as x, y, and orientation, using probability from LIDAR and odometry [4]. It is a renowned algorithm for indoor mobile robot localisation.

Path planning is another part of navigation in order to perform an autonomous inspection robot. In this research, the inspection robot deploys the robot operating system (ROS) as the central system. It provides several path planner packages to accomplish particular tasks. Each path planning algorithm has its benefits depending on usage. However, Goyal et al. [5] reveal that the most popular algorithms now are A* and Dijkstra, which offer faster and shorter paths than others. In addition, they conclude that A* is better than Dijkstra owing to the speed and computational burden. Even though there is not much information about path planning for the aircraft inspection robot, A* is commonly used in ROS for a typical mobile robot [6]. After considering A* as a global path planner for aircraft inspection, a local path planner should also be selected. The three practical algorithms are predictive

model control (MPC), Dynamic-Window Approach (DWA), and Timed Elastic Band (TEB). Ferrer Sánchez [7] examines these algorithms and proves that TEB is the best algorithm with regard to performance and collision avoidance. Naotunna & Wongratanaphisan [8] endorse TEB local planner as a better option for a differential drive robot than DWA and EBand, although DWA benefits smooth motion, time spent, and repeatability, while EBand offers high accuracy. Moreover, Rösman [9] concludes that TEB supports all robot types, multiple trajectories, and dynamic obstacles, but a significant computational burden is consumed, whereas DWA benefits diff-drive robots with no motion reversal and aids unsmooth cost function.

Turning to more details, particularly in TurtleBot3, de Assis Brasil et al. [10] evaluate global path planning between Dijkstra and A* with the same local path planner, DWA, for TurtleBot3 in the simulation that Dijkstra is slightly superior in the environment with the obstacle. Meanwhile, Cybulski et al. [11] assess three local plans: (1) DWA, (2) EBand, (3) TEB as the extension of EBand, by accuracy and repeatability in the simulation and real research platform. They used AR tags as a motion capture system to collect the data. In terms of results, DWA is notable for repeatability, EBand is remarkable for accuracy, and TEB is faster than others in the case of speed of action. In order to achieve the goal, a hangar is a large and open area with some dynamic obstacles, such as humans and moving vehicles. Therefore, A* could be suitable for the hangar because it can reduce computation and time spent and still offer an optimal path. In addition, A* can ensure receiving the ideal path with one processing [10]. That could influence task achievement and repeatability. Then, TEB could be used due to the capability of dynamic obstacle avoidance, speed of action, and accuracy, even though it has high computation. Hence, matching A* and TEB is purposed to compare with existing work from an original approach from the official TurtleBot3 used in this work [12]. Thanks to Zheng [13] and Mowito [14], some parameters and their effects in navigation packages are explained to tune suitable values for TurtleBot3 in several operating environments.

In this research, the executing robot was TurtleBot3, a popular mobile robot for learning and developing. Essential hardware is already allowed, namely Raspberry Pi, OpenCR board, Raspi camera, 2D LIDAR (LDS-02), and odometry sufficient for navigation and SLAM in an indoor environment. Also, it is a ROS-based mobile robot. Pyo et al. [3] affirm that ROS benefits developers in several aspects: firstly, it has a large active community, which has developed and shared over 5,000 packages and plenty of learning resources exceeding 18,000 pages from Wiki pages; secondly, it offers development tools such as debugging tools, 2D visualisation tool (rqt), 3D visualisation tool (RViz), and 3D simulator (Gazebo) that are necessary features for development; lastly, ROS provides excellent communication between hardware, software, and users.

As an equipped detecting sensor, LIDAR only performs obstacle avoidance. Hutabarat et al. [15] examine LIDAR-based obstacle avoidance for a mobile robot that can perfectly avoid a variety of colours and widths of obstacles with accurate measure distance, but the transparent objects are not able to detect due to no reflected laser pulses. On the other hand, LIDAR could not be suitable in all cases, depending on the situation and environment. The hangar has an open area involving an impact of high light intensity and some objects

lower than LIDAR's level. Therefore, relying on a LIDAR entirely will not be a smart option. According to Campbell et al. [16], the ultrasonic sensor offers robustness, reliability, and cheap cost by using sound wave to measure the distance to an object, although temperature and humidity changes in a medium is weakness. Therefore, the ultrasonic sensor becomes an alternative sensor to incorporate with LIDAR to perform collision avoidance. Not only its functionalities are in consideration but also size, computational burden, and power consumption are appropriate for TurtleBot3 waffle pi.

Researchers and developers have been fascinated by the use of the ultrasonic sensor in mobile robots. Azeta et al. [17] investigate obstacle detection using an ultrasonic sensor that can detect several types of objects without the impact of lighting conditions, but they found it hard to avoid a surface. Due to the lighting condition in the hangar, the ultrasonic sensor is still a wise selection for dealing with collision issues. Moreover, using a single ultrasonic sensor works well with various shapes of objects, including moving objects [18]. Moving objects are possible in aircraft inspection, such as operators, vehicles, and even other operating robots, so installing an ultrasonic sensor as a redundant sensor is better to ensure safety.

In the view of implementation, equipped sensors in the robot should detect surroundings and store the data to reach the goal with collision-free. A layered costmap integrates each type of tracking obstacle layer into the master layer for navigation, leading to faster planning and cleaner separation of concerns [19]. The `costmap_2d` package in ROS uses the layered costmap method to store and update sensors and static map information about obstacles in the environment for navigation [20]. This method and package are helpful in this research owing to the association with SLAM, LIDAR, and an ultrasonic sensor as multiple layers.

III. METHODOLOGY

To achieve the aim and objectives, the experimental design is a key to verifying navigation performance related to objectives. General standards and regulations for the indoor mobile robot should be the criterion for evaluating different navigation methods, such as path planning. Next, the required hardware and software will be layout to deal with immediate issues and implementation straightforwardly. Then, navigation will be designed to reach the goals, namely mapping, localisation, path planning, and obstacle avoidance. Finally, the system architecture will also be shown to ensure that communication and navigation are well-designed and convenient to modify when facing issues.

A. Experimental design

Regarding benchmark protocol for indoor mobile [21], challenges in a domestic environment for a smart city by SCIROC [22], and indoor mobile robot competition [23], the data collection form was designed to save the necessary information and compare navigation performance. Initially, the algorithm's name will be placed in the first column, which could be convenient to classify. Then, the time stamp in ROS can show the start and finish times, a reliable source similar to indoor mobile robot competition. Then, the start and goal coordinates show the travel route because there are several waypoints to be tested. Next, ROS can also inform the number of warnings and errors, so this study pays attention to unusual situations to be observed and recorded. Afterwards, failures will be counted and noted following the benchmark protocol.

Then, the time spent can be calculated by the difference between the start and finish time. Finally, positioning errors will be measured using tape measurement between the ground truth and the actual position. It was required five times for each test to calculate average values. Ground truth was required to be created first because the robot could start at the same position every time and reach determined waypoints without interruption.

Algorithm	Start time	Start coordinate	Finish time	Goal coordinate	Number of Warning	Number of Error	Number of failures	Distance to failure	Time to failure	Time spent	Positioning error
Average											

Fig. 1. Data collection form

The difference between the marking waypoint and heading is a positioning error based on the Euclidean distance equation and should be less than eighty centimetres [21].

Turning into the simulation in more detail, two destinations were set: (3.0, -0.5) and (3.0, 1.5). Basically, the robot used different approaches, Dijkstra with DWA and A* with TEB, to perform the test by five times per destination to compare navigation performance. The static map was created by SLAM, a controlled factor in all experiments. However, the moving box was unpredictable because it moved from point (0, -1) to point (0,1) as a loop with a velocity of 0.1 m/s. In order to run the test, a Python script was required to calculate generated path distance using the initial pose and move_base/goal topic [24]. The time was stamped when the robot started and ended. Also, all warning and error messages were recorded to observe unusual behaviours and problems. Noted that path distance was computed from a primary global plan; it was not an actual path that the robot travelled.

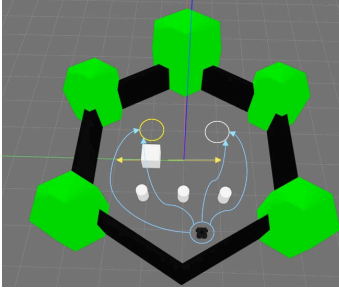


Fig. 2. Simulation test

The office has a 2.5 m x 1.0 m area. Then, three waypoints were defined: (1.0, 0.0), (0.0, -0.5), and (2.0, -0.5) to validate the navigation performance following objectives. Besides, the first waypoint is (1.0, 0.0), meaning that the target is in the x direction. Next, the second waypoint, (0.0, -0.5), is for the y direction, and the last waypoint is (2.0, -0.5), a combination of the x-y direction.

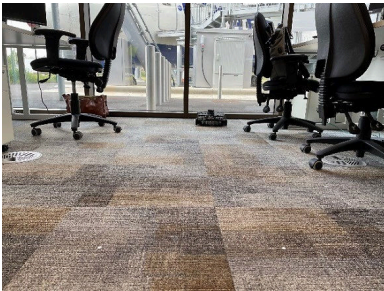


Fig. 3. An experiment in the office

The atrium was simulated as a hangar model to ensure that the system was ready to run before testing at the hangar. In the atrium, it was allocated around 3.5 m x 4.5 m of area, bounded by tables. Next, three waypoints were determined: waypoint 1 (2.5, -0.8); waypoint 2 (1.0, -2.55); initial waypoint (0.0, 0.0). The robot was designed to move from the initial waypoint to waypoints 1 and 2 and return to the initial waypoint. The initial waypoint was also marked as same as the office environment. However, the robot ran to all waypoints in this test with a delay on each point of about 20 seconds for marking the position.

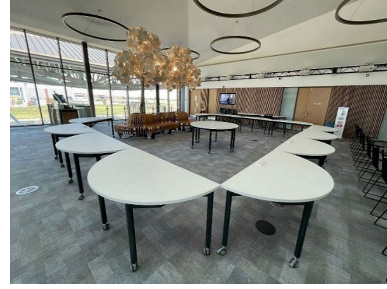


Fig. 4. An experiment in the atrium

The dimension of the hangar was provided by drawing so that the overall area is about 30m x 30m. The initial position was defined as 2m away from a pillar because it could be convenient for several replicating tests with a particular landmark. Then, The first checkpoint was determined at point (2.5, 0.0), which is in step 14 of the instruction for inspecting a wingtip, and the second checkpoint was at a point (5.9, 7.0) to check tire bolts following step 19. The mobile robot has to perform multiple waypoints, including the return home position, but it has 20 seconds delay at each waypoint, assuming an inspecting period.



Fig. 5. An experiment in the hangar

B. Robot system

Navigation indicates the robot's movement from start to destination, such as optimised route and collision avoidance [4]. The fundamental navigation requirements are map, pose of the robot, sensing, and path planning. It can be acquired by these methods: (1) Simultaneous Localisation and Mapping (SLAM) was applied to explore unknown areas, detect surroundings, and estimate the current location beneficial to generate a map; (2) Adaptive Monte Carlo Localization (AMCL) estimates the pose of the robot by gathering data from odometry, distance sensor (LIDAR); (3) Sensing installed in TurtleBot3 waffle pi is odometry, IMU, and LIDAR that are primary sensors using in the navigation; (4) Path planning is a significant indicator of navigation because each algorithm has its advantages and disadvantages depending on mission and environment. Owing to map-based navigation, SLAM is executed to receive the map. A laser-

based SLAM by gmapping technique was deployed in the mapping process. The gmapping is based on odometry and laser scan to obtain the pose and landmarks by probabilistic method, the Rao-Backwellized particle filter, sampling weight, and resample [25]. Typically, a map is required for an indoor mobile robot in all environments. SLAM is a standard tool for creating a map for the robot. It gathers the surroundings' data and distance between things into the map that the robot can use as a reference to perform navigation.

Path planning comparison is an essential outcome of this study. Various environments are used to verify the navigation performance of each method. The first approach is Dijkstra with DWA, offered by the official TurtleBot3 [26]. Another approach is A* with TEB, provided by ROS Planning [27] and TU Dortmund [28]. Global and local planners ensure an effective plan to reach the goal.

As to the use of A* in ROS, ROS Planning [27] provides a navigation stack package together with a grid path and quadratic calculator. It is a general package for a mobile robot. In order to execute the package in TurtleBot3, it needs to install the package by ROS or clone it from GitHub to a workspace. Then, tuning parameters in move base, global costmap, local costmap, and common costmap is required with respect to TurtleBot3 identification. The results of the parameters will be demonstrated in the notable parameters section.

A local planner is the second plan when the robot deals with unseen objects on the map and dynamic obstacles. It could support global planners in reaching the goal efficiently. TEB could optimise the trajectory generated by the global planner and corporate with the robot under ROS by teb local planner package [28]. TEB package is required to clone from GitHub and then add it in move base launch file. However, this package is for mobile robots in general. Tuning parameters are necessary to navigate the robot smoothly and correctly in a particular environment. The more complicated situations are, the more sophisticated adjustment is required.

Collision-free is another regulation to achieve the goal; if the robot is not able to avoid obstacles, it will get stuck around the obstacle resulting in failure to reach the goal. Regarding LIDAR's weakness, more perceptions should be added that it is an ultrasonic sensor. It is not only a basic range sensor but also a low price. HC-SR04 was installed in front of the robot owing to the moving forward-only function. In view of connection and coding, the sensor consumes power from OpenCR board, and then Trig and Echo ports connect to GPIO_1 and GPIO_2, respectively. Afterwards, to receive data from the sensor, the ultrasonic sensor's code based on equation 2 must be integrated into turtlebot_core, compiled and uploaded by Arduino IDE. In addition, the sensor's data have to be published into ROS, that is /sonar topic. Although OpenCR is compiled with C, it can communicate with ROS flawlessly by rosserial package.

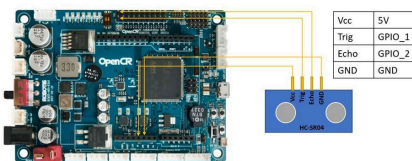


Fig. 6. A connection between OpenCR [26] and HC-SR04

Thanks to the layered cost map, LIDAR and the ultrasonic sensor perform together and mark its detection to its layer to generate the master layer for navigation. In other words, the master layer is a combination of a static map layer created by SLAM, an obstacle layer built by LIDAR, a sonar layer generated by an ultrasonic sensor, and an inflation layer produced by boundary setting. The environmental data is contained in a layered costmap to execute path planning and obstacle avoidance.

In terms of use, a navigation package is needed to clone it into the workspace. Ferguson [29] provides costmap layers inside the navigation package, ready to use. The desired layers have to add to the local costmap layer and global costmap layer. Also, the layers and specific parameters must be inserted into common costmap params file. In this work, the deployed layers are the static map layer, inflation layer, obstacle layer, and sonar layer to perform obstacle avoidance.

IV. RESULT

A. Simulation

Regarding the results shown in the charts, Dijkstra with DWA spent more time than A* with TEB for both waypoints. Although Dijkstra is the shortest path, it generated a path longer than A* with TEB. The reason is that Dijkstra reserved distance between the robot and obstacles farther than A*, and longer length also affected time spent. In positioning error, Dijkstra with DWA was better than A* with TEB, but it had one failure in waypoint 2 that was far more error, leading to a high average positioning error. The failure was caused by a moving box that occupied the path frequently until the robot could not find a way to the destination.

Meanwhile, the time could not be compared remarkably because of the uncontrolled moving box. For instance, if the moving box is going to block the path, the robot will find a new path resulting in a long distance and time spent. In contrast, if the robot is going out from the occupied space, the robot will update and replace a new path into that space leading to a shorter distance and time. It could estimate the system's running time in general, and the two methods have no significant differences.

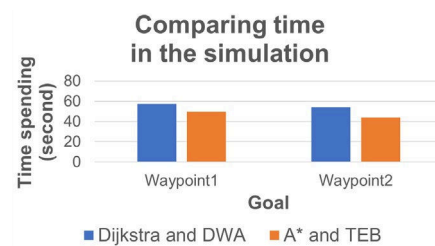


Fig. 7. Chart comparing time in the simulation

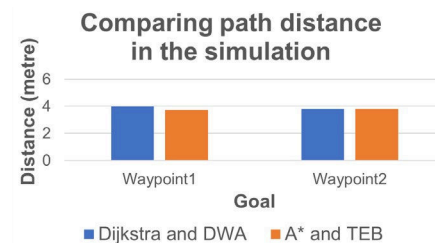


Fig. 8. Chart comparing path distance in the simulation

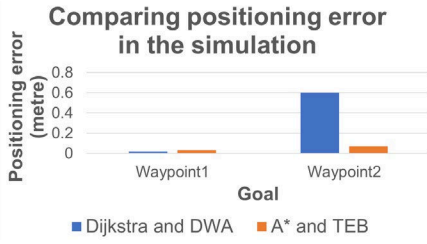


Fig. 9. Chart comparing positioning error in the simulation

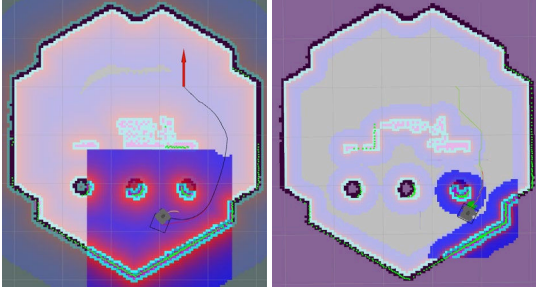


Fig. 10. Waypoint 1 in the simulation, Dijkstra with DWA (left) and A* with TEB (right)

As an aspect of objectives, Dijkstra with DWA achieved the task four out of five times, with a repeatability and success rate of eighty per cent. At the same time, A* with TEB had no failures, so it could be marked as an excellent record. Nonetheless, it could not conclude that A* with TEB is better than Dijkstra with DWA because it still has more environments to test, and the aimed target area is the hangar. At least these two approaches did not hit any objects in the experiment. Then, the accuracy could be evaluated as consistent as the positioning error that Dijkstra had better accuracy in waypoint 1, and worse than A* with TEB in waypoint 2.

B. Office

In the point of results, the charts show that Dijkstra with DWA spent less time while A* with TEB provides a few positioning errors in all directions. Intriguingly, the robot had a positioning error in the y direction higher than the x direction in both methods despite using the same localisation method, AMCL. Although the office decorates with furniture, it has open spaces, such as under the desks, corridor, and a glass wall. Also, chairs will be moved, so the chairs on the map cannot be in the same position as realistic, influencing deviated landmarks in localisation. Therefore, LIDAR could not provide accurate localisation as its performance should. The robot then relied on odometry, which could generate an accumulated error. In this case, the error occurred remarkably in the y direction. The number of warnings and errors was nearly the same.

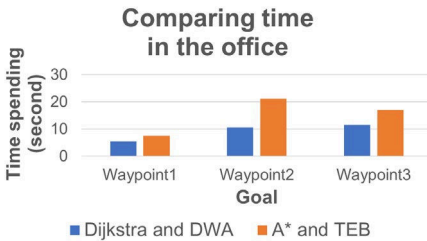


Fig. 11. Chart comparing time in the office

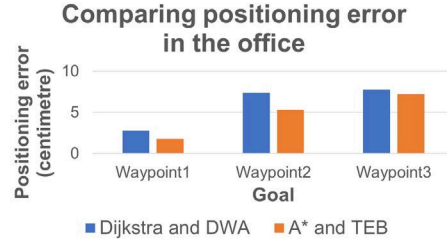


Fig. 12. Chart comparing positioning error in the office

In the matter of objectives, the two methods could complete the task by reaching all desired destinations. Each test was examined at least five times, which could signify the system's repeatability. After installing an ultrasonic sensor on the robot, the robot could avoid the chair's legs in the office and no collisions anymore during the experiments. In terms of accuracy, A* with TEB performed better than Dijkstra with DWA moderately.

C. Atrium

As to the result, the charts are illustrated to make a clear comparison. In waypoint 1, A* with TEB spent more time than Dijkstra with DWA almost two times, but the positioning error was less than four times. Similar to waypoint 2, A* with TEB was good at accuracy, although it took much time. On the other hand, time spent is equivalent in the return home position, and A* with TEB still provided better accuracy in positioning.

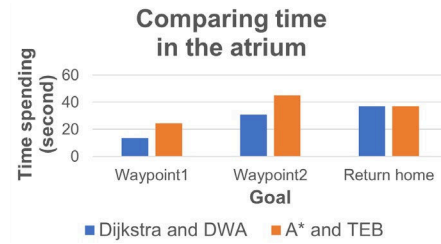


Fig. 13. Chart comparing time in the atrium

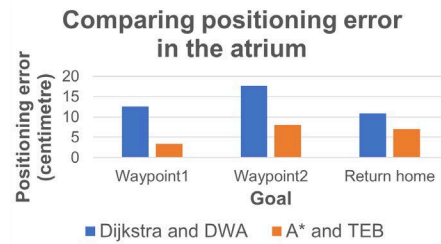


Fig. 14. Chart comparing positioning error in the atrium

To analyse the results relevant to the goals, these two approaches did not fail and exceeded positioning error requirements as a consequence of task achievement. Likewise, repeatability could be capable because of collecting data in several tests, and the outcomes did not differ much. The robot also performed collision-free in all tests. The results pointed out that A* with TEB offered higher accuracy than Dijkstra with DWA.

D. Hangar

Turning to the results, in the first waypoint, Dijkstra with DWA was slightly more accurate than A* with TEB, but the time spent was almost the same. Although the time spending was not substantially different in the second waypoint, Dijkstra with DWA had positioning error excess regulation

that was unacceptable. Lastly, the robot needed to return home position. A* with TEB was moderately greater than Dijkstra with DWA in both aspects.

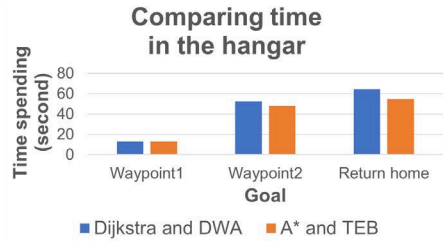


Fig. 15. Chart comparing time in the hangar

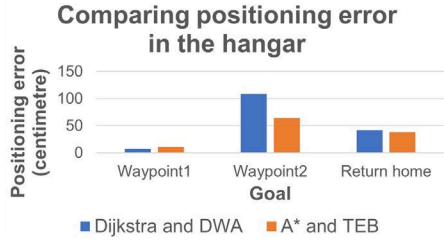


Fig. 16. Chart comparing positioning error in the hangar

Indeed, A* with TEB can accomplish determined multiple waypoints in the hangar, whereas Dijkstra with DWA fails in waypoint 2 due to higher error than regulation, indicating poor accuracy too. No crashes were found in the experiment. The experiment was run at least five times to compute average values, implying that the system has repeatability.

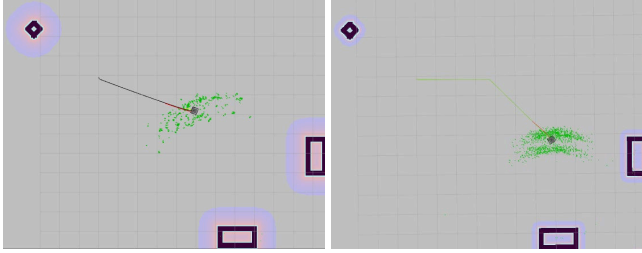


Fig. 17. Waypoint 2 in the hangar, Dijkstra with DWA (left) and A* with TEB (right)

Interestingly, Each method showed its outstanding path following its principle. It illustrates that Dijkstra was the shortest path, but A* was an optimal path using the Best-First search. However, the robot's localisation was extremely terrible because of vastly scattered particles.

E. Obstacle avoidance

Unfortunately, LIDAR detects free spaces as obstacles around the robot in the high light intensity area in spite of no objects in that area, influencing navigation. So, the obstacle layer, using LIDAR data, must be closed to maintain the goals. Then, the ultrasonic sensor becomes the main sensor for obstacle avoidance. Even though the hangar has few objects on the ground, obstacle avoidance should be tested for safety. A tape measure was placed in front of the robot to interrupt the path, but the robot could find the tape measure and replan to avoid it properly. Hence, the robot could perform collision-free in all scenarios following the objectives.

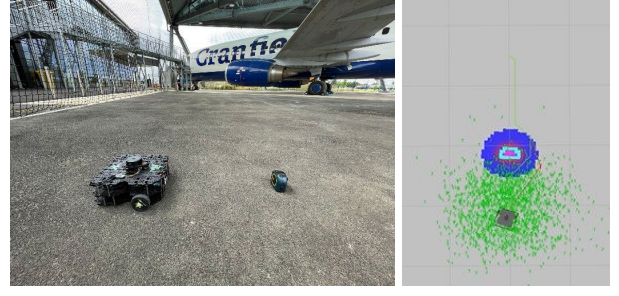


Fig. 18. The obstacle avoidance in the hangar

F. Notable parameters for TurtleBot3

The A* and TEB are the main packages implemented to TurtleBot3 because Dijkstra and DWA are official packages that were tuned already for commercial purposes. After understanding the parameters and their effects, the values were adjusted by TurtleBot3 specifications and trial and error in the experiments.

TABLE I. COSTMAP PARAMS

Costmap params		
Parameters	Value	Details
inflation radius	0.5	It is a buffer zone around obstacles.
footprint clearing enable	True	It should be True, in order to clear and update the footprint.
cost scaling factor	20	It is an exponential rate in obstacle cost. The robot will find a path between obstacles properly.
clear_threshold	0.4	It is a probability to clear obstacles from the map.
mark_threshold	0.8	It is a probability of marking obstacles on the map.
clear sensor cone	True	It is to clear objects out of the map when the area is updated as free space.

TABLE II. TEB LOCAL PLANNER PARAMS

TEB local planner params		
Parameters	Value	Details
dt_ref	0.3	It is a resolution of trajectory. The small value is fine resolution resulting in an excellent approximation, but the huge value leads to a not feasible path.
dt_hysteresis	0.1	It is a redundant resolution of trajectory. It could extend dt_ref within $\pm dt_hysteresis$.
footprint model	Point	To define the dimension of the robot. The robot will be assumed as a particle if a point value is used.
xy goal tolerance	0.05	Minimum error in x and y goal position. Too much value leads to poor accuracy, but a small value results in oscillation and not reaching the destination.
yaw goal tolerance	0.05	Minimum error in orientation. A high value provides many errors, whereas a low value generates oscillation and failure.
min obstacle dist	0.15	Minimum distance between the robot and obstacle. It depends on the environment and spaces.

V. DISCUSSION

In performance verification, some parameters cannot be measured, including the orientation error and the velocity. Though the positioning error can be quantified, it is not an appropriate technique. In other words, using a tape measure is a simple method of obtaining a range, but it could not provide exact values due to misreading and deviation by human error.

VI. CONCLUSION

Aircraft maintenance is a combination of several processes, including a preflight inspection. To support the inspection, navigation is a primary unit for a mobile robot to achieve. This study shows that TurtleBot3 can move autonomously to defined waypoints: (1) wingtip check; (2) tire bolt check; (3) return home in the hangar. In terms of objectives, the robot can achieve the desired tasks in various environments as well as the hangar. Also, it was tested several times to confirm repeatability. Next, the robot can avoid potential obstacles, especially static obstacles, thanks to LIDAR and an ultrasonic sensor. Although it could avoid simple dynamic objects, it was successful only in the simulation. Then, the robot achieves good accuracy where AMCL acts on the environment efficiently, but accuracy is down in the hangar due to using odometry mainly as a localisation, which should be improved further. Both path planning approaches have their advantages, and A* with TEB is more appropriate for navigation in aircraft inspection at the hangar. The mobile robot could support operators and deal with human errors to increase quality and productivity, resulting in safety and effective cost.

In future work, an outdoor mobile robot should be used due to its size and outdoor LIDAR type to cover the entire area in the hangar. Moreover, localisation should be improved by using markers, machine vision techniques, or real-time location systems.

ACKNOWLEDGMENT

I sincerely thank Cranfield University, Digital Aviation Research and Technology Centre (DARTEC), and my supervisors for giving me opportunities and resources.

REFERENCES

- [1] Federal Aviation Administration, "13-1B - Acceptable Methods, Techniques, and Practices - Aircraft Inspection and Repair," United States Department of Transportation, 1998. https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_43.13-1B_w-chg1.pdf
- [2] L. Andre, "EXTERIOR INSPECTION B737-800," 2001. <https://www.yumpu.com/en/document/view/11406964/exterior-inspection-b737-800-japstorecom> J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
- [3] R. Zegers, "Path Planning Basics," The Construct, 2022. <https://app.theconstructsim.com/Course/70>
- [4] Y. Pyo, H. Cho, I. Ryuwoon, J. I. and T. Lim, "A Handbook Written by TurtleBot3 Developers Robot Programming From the basic concept to practical programming and robot application," 2017. [Online]. Available: www.robotis.com
- [5] A. Goyal, P. Mogha, R. Luthra, and N. Sangwan, "Path finding: A* or dijkstra's?," *International Journal in IT & Engineering*, vol. 2, no. 1, pp. 1–15, 2014.
- [6] T. Schwörer, "global_planner," *Open Robotics*. http://wiki.ros.org/global_planner (accessed July 2022).
- [7] J. Ferrer Sánchez, "Implementation and comparison in local planners for Ackermann vehicles," Post graduate theses, Dept. Ind. and Inf. Eng., Politecnico di Milano, Italy, 2018. [Online]. Available: https://www.politesi.polimi.it/bitstream/10589/141744/3/Tesina_JordiFerrer.pdf
- [8] I. Naotunna and T. Wongrataphisan, "Comparison of ros local planners with differential drive heavy robotic system," in *2020 International Conference on Advanced Mechatronic Systems (ICAMEchS)*, 2020, pp. 1–6.
- [9] C. Rösmann, "Difference between DWA and TEB local planners," Jun. 2022. [Online]. Available: http://files.davidqu.com/research/tutorials/2018_rosmann_Difference%20between%20DWA%20and%20TEB%20local%20planners.pdf
- [10] P. M. de Assis Brasil, F. U. Pereira, M. A. de S. L. Cuadros, A. R. Cukla, and D. F. T. Gamarra, "A study on global path planners algorithms for the simulated turtlebot 3 robot in ros," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, 2020, pp. 1–6.
- [11] B. Cybulski, A. Wegierska, and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot," in *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, 2019, pp. 104–111.
- [12] H. Yu, "Monitoring Robotic Vehicle Movements Using Precision Location System," Cranfield University, 2021.
- [13] K. Zheng, "Ros navigation tuning guide," in *Robot Operating System (ROS)*, Springer, 2021, pp. 197–226.
- [14] Mowito, "Step 5-c: Controller - TEB Local Planner Tuning Guide," Mowito. https://mowito-navstack.readthedocs.io/en/latest/step_5c.html (accessed July 2022).
- [15] D. Hutabarat, M. Rivai, D. Purwanto, and H. Hutomo, "Lidar-based obstacle avoidance for the autonomous mobile robot," in *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, 2019, pp. 197–202.
- [16] S. Campbell *et al.*, "Sensor technology in autonomous vehicles: A review," in *2018 29th Irish Signals and Systems Conference (ISSC)*, 2018, pp. 1–4.
- [17] J. Azeta, C. Bolu, D. Hinvi, and A. A. Abioye, "Obstacle detection using ultrasonic sensor for a mobile robot," in *IOP Conference Series: Materials Science and Engineering*, Dec. 2019, vol. 707, no. 1. doi: 10.1088/1757-899X/707/1/012012.
- [18] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghighbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Low-cost ultrasonic based object detection and collision avoidance method for autonomous robots," *International Journal of Information Technology*, vol. 13, no. 1, pp. 97–107, 2021.
- [19] D. v Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 709–715.
- [20] N. Lamprianidis, "costmap_2d," *Open Robotics*. http://wiki.ros.org/costmap_2d (accessed July 2022).
- [21] C. Sprunk *et al.*, "An experimental protocol for benchmarking robotic indoor navigation," in *Experimental Robotics*, 2016, pp. 487–504.
- [22] SCIROC (2020), "ERL Consumer Rulebook Update," https://eu-robotics.net/robotics_league/upload/erl-rulebooks/ERL_Consumer_2020_Updated.pdf (accessed June 2022).
- [23] DARPA Grand Challenge (2007), "Urban challenges rules," https://www.grandchallenge.org/grandchallenge/docs/Urban_Challenge_Rules_102707.pdf (accessed June 2022).
- [24] G. Stavrinou, "navigation_tools," *GitHub*, 2022. https://github.com/gstavrinou/navigation_tools (accessed Aug. 2022).
- [25] Y. Cheng and G. Y. Wang, "Mobile robot navigation based on lidar," in *Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018*, Jul. 2018, pp. 1243–1246. doi: 10.1109/CCDC.2018.8407319.
- [26] ROBOTIS, "TurtleBot3," *GitHub*. <https://github.com/ROBOTIS-GIT/turtlebot3> (accessed July 2022).
- [27] ROS Planning, "Navigation," *GitHub*. <https://github.com/ros-planning/navigation> (accessed July 2022).
- [28] TU Dortmund, "teb_local_planner," *GitHub*. https://github.com/rst-tu-dortmund/teb_local_planner (accessed June 2022).
- [29] M. Ferguson, "ROS Navigation Stack," *GitHub*. <https://github.com/ros-planning/navigation> (accessed July 2022).



DSPACE

<https://dspace.org/>

Navigation for a mobile robot to inspect aircraft

Mansakul, Thanavin; Fan, Ip-Shing; Tang, Gilbert

2023-08-08

Institute of Electrical and Electronics Engineers (IEEE)

<https://doi.org/10.1109/YEF-ECE58420.2023.10209290>

Mansakul T, Fan I-S, Tang G. (2023) Navigation for a mobile robot to inspect aircraft. In: 2023 7th International Young Engineers Forum (YEF-ECE), 7 July 2023, Lisbon, Portugal