International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 10 Issue 5, Sep-Oct 2026 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

A Hybrid Sensor-Range Exploration Framework for Autonomous Mobile Robot Map Creation

Shamsudeen A Sodangi, Professor Liu Hui, Associate Prof. Dr. Li Xueyuan

Department of Mechanical Engineering, Federal Polytechnic, Nasarawa, Nigeria

ABSTRACT

Autonomous map creation in unknown environments remains a core challenge in mobile robotics. While traditional frontier-based and sampling-based exploration strategies efficiently cover space, they frequently induce redundant paths, unnecessary rotational manoeuvres—particularly in open workspace domains. This paper presents a hybrid exploration framework that combines a global structured path inspired by Complete Coverage Path Planning (CCPP) with a local boundary-following mechanism. Unlike standard CCPP algorithms that require the physical chassis footprint to sweep every unit of area, the proposed algorithm requires only that the effective sensing range bubble of the robot sweeps the workspace. When physical obstacles obstruct the sensor range bubble and prevent line-of-sight coverage, a secondary local wall-following routine is triggered to circumnavigate the obstacle. Kinematic equations for linear and rotational execution times are derived from first principles based on trapezoidal and triangular speed profiles. The framework was implemented in ROS and evaluated in Gazebo against Frontier Exploration and RRT Exploration baselines using differential-drive (Kobuki) and skid-steer (Husky UGV) platforms. Results show that the hybrid sensor-bubble approach reduces path repetitions during exploration while achieving complete map coverage on different robot platforms.

KEYWORDS: *Autonomous Mobile Robots, Map Creation, ROS, Sensor Range Bubble, Path Planning, Wall-Following, CCPP.*

How to cite this paper: Shamsudeen A Sodangi | Liu Hui | Li Xueyuan "A Hybrid Sensor-Range Exploration Framework for Autonomous Mobile Robot Map Creation" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-10 | Issue-5, October 2026, pp.1375-1382, URL: www.ijtsrd.com/papers/ijtsrd142327.pdf

IJTSRD142327

Copyright © 2026 by author (s) and International Journal of Trend in Scientific Research and Development Journal. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0) (http://creativecommons.org/licenses/by/4.0)

INTRODUCTION

Map creation and autonomous navigation in unstructured or unknown environments are critical for applications such as industrial inspection, agricultural monitoring, and search-and-rescue operations. These maps are used to create the most efficient trajectory that minimises total path length, energy consumption, and operational execution time. Exploration refers to the translation and rotation of the mobile base through an unknown environment such that a complete map of the environment is created from the onboard sensors [1].

Classical exploration algorithms, such as Frontier Exploration, identify boundaries between known free space and unexplored areas, directing the robot toward the nearest or most informative boundary [2]. Similarly, sampling-based techniques like Rapidly-exploring Random Trees (RRT) dynamically expand search trees to identify target exploration points [3]. However, both approaches are prone to path oscillations, repeated backtracking across previously mapped areas, and excessive rotational adjustments when transitioning between distant exploration areas.

To address these limitations, this paper introduces a hybrid exploration framework. An initial path is drawn in the exploration area that assumes the map is free of obstacles, this step is similar to traditional Complete Coverage Path Planning (CCPP). The difference is rather than requiring the full physical chassis footprint to translate through the exploration area, only an expanded bubble that encloses the sensor range is required to translate through the map. When an obstacle intersects this sensor bubble and occludes perception, the framework transitions to a secondary local boundary-following mode to circumnavigate the obstruction.

Robot Exploration Strategies

Frontier cells approach is the most prominently featured type of algorithm in the literature of robot exploration. The basic mechanism works by identifying the boundary between explored regions and unexplored regions of the defined exploration area. Points on the boundary are marked as "frontier cells". The mobile base is despatched to these "frontier cells" based on some selection criteria. This method is first reported in the literature in 1998 by Yamauchi in [4]. Several modifications have since been proposed and implemented in several frameworks including by [2], [5], [6]. Generally, frontier cells solutions generate a series of goals for a mobile base to achieve based on local information. Biasing the selection process with criteria like expected information gain can be very effective, and these algorithms perform well as shown in the comparisons by [7]. Other approaches integrate frontier cells as a secondary module to support a more primary behaviour. An example of this was presented by [8]; the system was designed to improve map quality by creating a path that returns to a previously scanned position (closed loop).

A frontier cells approach was presented by "Makarenko et al." in [9]. A utility function that considers the information gain, navigation cost and localizability was used to decide which frontiers to explore. The standout metric in this approach is the use of localizability, defined as the probability the vehicle can localize itself at target location. While "Ström et al." in [10] also employs a frontier cells approach that uses loop-closing and re localization actions to choose between unexplored frontiers. The system was particularly designed for mapping underground structures, hence most of the space is inaccessible. This makes loop-closure particularly important. "Kollar & Roy" present an approach that uses machine learning to generate smooth trajectories that decrease the uncertainty in localization as was presented in [11]. The core attribute of the reinforcement learning parameter is a bias towards controls that minimize errors in the robots odometry.

Coverage Problem

CCPP has been extensively studied in robotics for mapped and unmapped environments. CCPP is also referred to as path planning of coverage region (PPCR) in the literature. "(CCPP) is the problem of finding a path that passes through all the points in the workspace from a starting point to a final point while avoiding obstacles" [12]. Applications of coverage path planning are usually applied to mine sweeping, lawn mowers and vacuum cleaners. Algorithms are generally optimized for minimum distance in the exploration space for holonomic mobile bases. Solutions for non-holonomic mobile bases may consider minimising rotations. CCPP offers good mathematical models for generating paths that lead to coverage in known environments. CCPP algorithms are generally designed to minimise time and/or distance for complete coverage of a given space. Figure 1 shows a typical path generated by a CCPP algorithm [13].

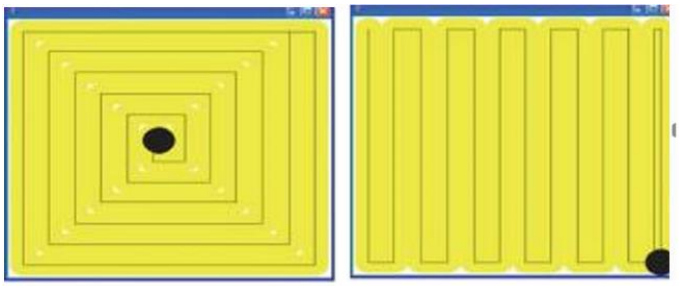


Figure 1: Typical CCCP Path [13]

Robot exploration may be modelled as CCPP problem by taking the robot sensor range as the footprint of the robot. In a completely empty environment or a mapping exploration that is conducted with a flying robot, that is perfectly valid and optimal for minimizing the path length. However, in an area populated with obstacles simply following such a path will mean some parts of the map will not be scanned as the sensor is obstructed by an obstacle, an obstacle might also be directly on the path. This work therefore uses this path only as a strong bias for exploring different paths of the map. When an obstacle obstructs points that would have been otherwise scanned following the CCPP model, the path will need to be modified to scan areas otherwise missed. The actual algorithm in full is described in the next section.

Algorithm Description

This section describes a new algorithm designed to achieve autonomous exploration of an environment autonomous for map creation. The algorithm is designed to assign goal points to a mobile base equipped with a distance sensor. The sensor is required to have a minimum 180 degrees coverage for complete exploration. The algorithm flow chart is shown in figure 2, and the remainder of this section is used to describe the algorithm in detail, with supporting equations and geometric analysis.

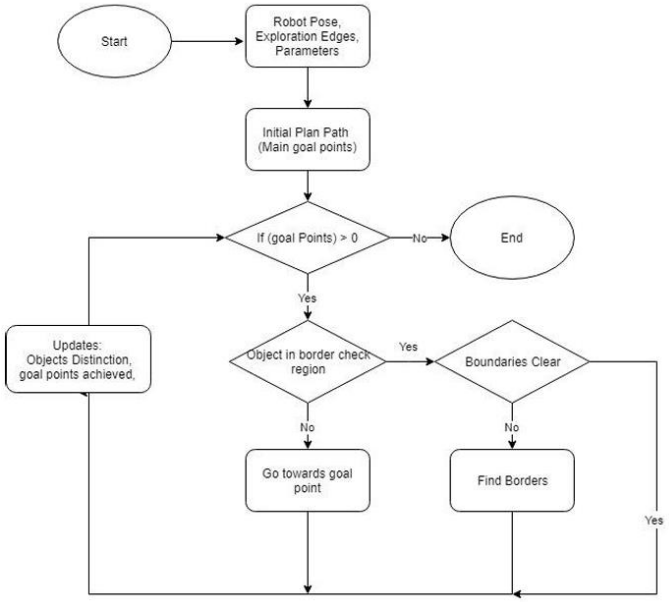


Figure 2:Algorithm Flow Chart

The algorithm will generate a path as shown in figure 3. The algorithm is initialized by selecting the vertices that define the area to be explored. This area is shown by the yellow boundary in this example. The exploration area is then divided into segments where the first segment is built by offsetting the exploration boundary (in yellow) inwards at a distance equal to twice the sensor range. These offset segments are continually created until the offset distance is too large to create another segment. The initial robot position is then used to generate the initial path (shown in white), this path is such that it explores between the drawn segments.

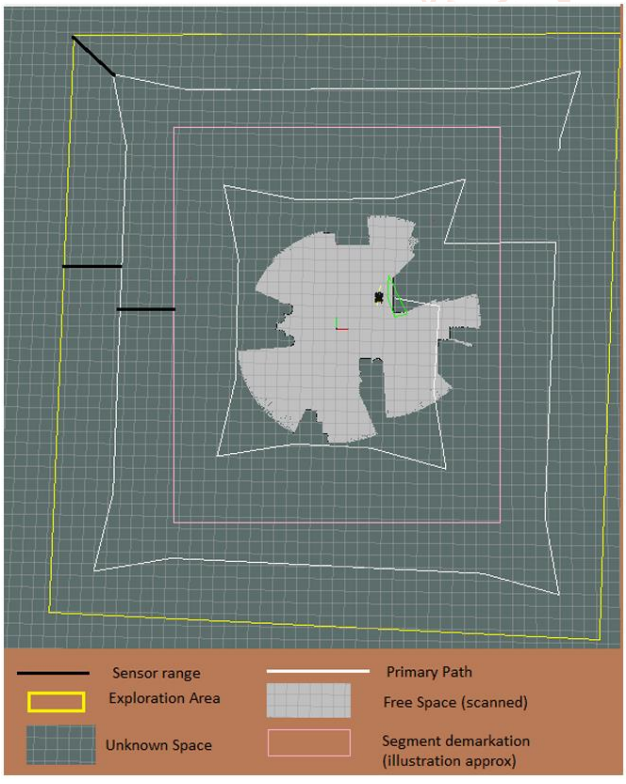


Figure 3: Initial Path that guides robot to exploration

There are two parts to the algorithm; calculating and generating the initial plan path and updating the plan as the robot moves along. The initial path is a series of goal points designed such that it is the path that takes the minimum time to traverse assuming the space is empty. The time of translation through any series of points on a 2D map is given by equation 1.

$$T = \left[\frac{1}{V} \times \sum_{i=0}^{i=n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}\right] + \left[\frac{1}{\dot{\theta}} \times \sum_{j=0}^{j=m-1} (\theta_j - \theta_{j+1})\right] \quad \ldots (1)$$

T is the total time to traverse the path, V is the linear velocity, n is the number of points in the path, x & y cartesian points, $\dot{\theta}$ is angular velocity, m is the number of rotations required, θ is angular displacement.

From equation (1), the linear velocity, and the angular velocity are functions of the robot base. To minimize the exploration time regardless of robot dynamic constraint, the length of the path and the total rotation is simply to be minimized. This evokes solutions provided in the CCPP literature. However, the difference of the footprint (in CCCP) of the robot, and the 'sensor bubble' (in this work) sweeping across the map to achieve complete exploration as discussed earlier requires corners be treated differently. The designed algorithm thereby treats corners as shown in figure 4.

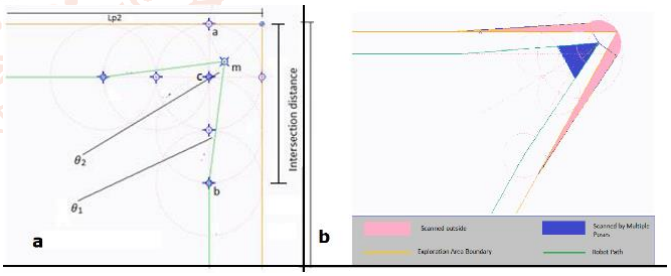


Figure 4:Path Generation Around Edge

From figure 4 (a), as the robot approaches point b the sensor range just reaches the wall on the right - where the sensor range is the circle. If the robot were to continue to point c then take a 90 degree turn, parts of the map will remain unexplored as shown by the circle drawn with center at c. Instead, the algorithm designs a new path as shown by the segment b-m. Figure 4 (b) shows that the entire section through the corner will now be explored, any unexplored area on the inner part of the area to be explored will be explored during the inner pass. This part is designed to offer the minimum deviation from CCCP thereby minimizing travel time as shown earlier in equation (1).

The second part of the algorithm deals with the following:

i. Obstacle on the path: When an obstacle is detected on the initial path, the robot switches to a wall following behavior. It circumnavigates the obstacle at a distance equal to the sensor range. This behavior (circumnavigation) is ended once the perimeter of the object has been fully mapped. Robot thereafter returns to following initial path.

ii. Obstacle between the robot and the edges of the segment to be explored: Similar behavior as described with obstacle on the path. When the object straddles the boundaries of the exploration area, only the perimeter within the exploration area is accessed.

iii. Goal point completely enclosed by an obstacle: Whenever a goal point is located squarely inside the perimeter of an object (obstacle) it is removed from the path and considered explored.

iv. Goal point reached: Once the robot is detected within a short distance of any goal point, the goal point is considered explored and removed from the list. This short distance is set at 10% of the range of the sensor.

v. Complete exploration: The exploration is considered complete when all the points in the initial path have been explored.

Algorithm Implementation

Several external libraries were used for access to special data types rather than functions. These type definitions have been accepted by the ROS community as the standard and are used in this implementation. These data types are available via ros_msgs and therefore individual classes are not required to be instantiated. Points and poses are represented in geometry_msgs, while paths and maps in nav_msgs format. The use of external libraries is not limited to these four examples.

Implementation of the algorithm was instantiated as one ROS node called "explore_node" and one external class "generate_loops". "generate_loops" is used to generate the initial path. 'explore_node' manages how the robot translates through the environment using the initial path and information gained.

This class "generate_loops" is called by the explore node, taking as argument the edges used to define the exploration boundary. 'generate_loops' is not a ROS node and is only constructed when the vertices are submitted. It draws out the path as detailed in the algorithm description.

The main function that manages the thread in explore_node is "manage_plan". "manage_plan" considers the progress of the main path and the obstacles on the map to call other functions. There is a while loop in "manage_plan" that tracks the progress of the robot in traversing the initial path generated. The goal poses on the path are removed based on robot proximity. The algorithm concludes once all goal poses are removed from the path.

A subscription to the local_costmap is used to develop a a record of obstacles. The local_costmap returns an occupancy grid around the robot of set size, and in the co-ordinate frame of the robot pose. Using an inbuilt sorting function, points detected as obstacles in grouped based on proximity into distinct individual obstacles called "features". Each feature is expected to represent a distinct continuous object in the real world. For computational efficiency the resolution (gap between points) of the features may be changed as a parameter based on computing power and environment size.

The implementation only sets goals for an external path planning algorithm like 'move_base'. As the robot approaches a goal another goal further on the path is selected, thus the robot does not slowdown in anticipation of stopping at the goal. To ensure the robot is properly aligned on the path, the robot is then required to achieve the pose (including orientation) within the set tolerance before continuing on the path if another behavior like obstacle circumnavigation has been triggered. This is also the case when the robot is returning from a boundary search to return to the path.

A session is shown in figure 5. In this instance the screenshots are arranged from left-to-right, and top-to-bottom. The robot is starting in the middle of the area to be explored. IT attempts to follow the initial path plotted while still ensuring the general area around the initial path it follows is fully explored before proceeding along the path.



Figure 5: An instance of explore borders (this work)

Testing Methodology

The algorithm developed in this work named "explore borders" was used to explore several environments running on different mobile platforms. The environments and mobile bases were simulated on ROS – GAZEBO. Two other exploration algorithms were simulated in the same environments, on the same mobile bases to serve as performance yardsticks.

A secondary node "measure_distance" was programmed to continuously record the change in poses of the robot and reports the sum every minute or upon request. This node is used to compare the algorithm developed in this work with others.

The following digital tools and works were used for testing this work:

i. **Gazebo**: "Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments" [14]. Gazebo is a standalone project that exposes its functionality to ROS via "simulator_gazebo".

ii. **Kobuki**: "TurtleBot 2 is an open robotics platform designed for education and research on state of art robotics." [15]. Kobuki is the mobile base attached to the very popular turtleBOT. Kobuki is equipped with a power source, motors and a sensor. A simulated kobuki mobile base was used to run exploration algorithms in this work.

iii. **Husky**: "Husky is trusted by hundreds of researchers and engineers globally. Numerous research papers have been published using Husky as the test set-up. Husky provides a proven benchmark for establishing new robot research and development efforts" [16]. Husky is a robot platform developed by Clearpath Robotics. The simulated Husky robot used in this work is made available by Clearpath Robotics via several opensource ROS packages.

iv. **rrt_exploration**: This is a popular work open sourced on ROS designed as an implementation of frontier cells exploration. It was designed and implanted by [17] as described in the work. Its popularity in the ROS community makes it a good benchmark to test this work against.

v. **frontier_exploration**: This is another popular opensource work on ROS using the frontier cells approach to map exploration. frontier_exploration was inspired by the work done by [4] and implemented by Clearpath Robotics.

Traditional measures of performance for exploration algorithms are; time taken, total translation, total rotation and the quality of map created. A series of experiments were carried out on a windows 10 computer with an Intel(R) Core(TM) i7-6820EQ CPU @ 2.80GHz, 2808 MHz, 4 Core(s), 8 Logical Processor(s). ROS was installed on ubuntu 16.04 running on VMware Workstation 14. Exact specifications of the setup for the experiments including a link to the package and launch files are provided in the appendix. Several simulations were performed and organized as two experiments presented below. All the algorithm-robot combination contrasted here have a high incompletion rate for the environments they were tested in. Some combinations of robot and algorithm had 100% incompletion in several trials and are therefore not considered. This is due to the probabilistic nature of localization algorithm and dynamics of the robot. Robot often gets stuck and becomes unable to maneuver out, there are also inevitable problems with the parameters required for robot, planner, and controller to work seamlessly especially in challenging environments.

Results

The map is generated by an external class (gmapping), map quality is therefore not a function of this algorithm. The time taken is also a function of robot dynamics and the underlying external controller. The fundamental property considered is therefore the translation and rotation of the robot.

Experiment 1:

An environment with random obstacles littered was used. This is shown in figure 6. Figure 6 is an instance of frontier exploration. The results are shown in table 1 below.

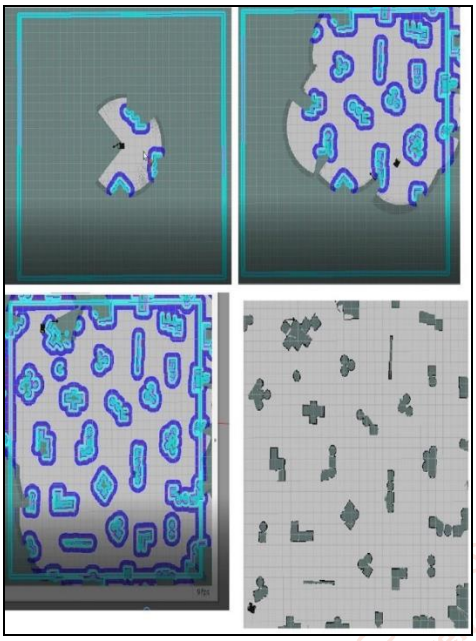


Figure 6: Screenshots of frontier exploration during experiment 1

Experiment 2:

A different environment with random obstacles littered was used. This is shown in figure 7. The results are shown in table 2 below.



Figure 7: Simulated Environment Used in Experiemnt 2

Table 1: Results of experiment 1

Platform	Algorithm	Trial 1 (D (m)/ R (rad))	Trial2	Trial 3	Mean
Husky	Explore Borders (This work)	172/242	142/207	138/191	151/213
Husky	Frontier Exploration	150/139	144/165	-	147/152
Kobuki	Explore Borders (This work)	241/594	231/539	196/507	214/547
Kobuki	RRT	238/519	209/253	277/201	241/324

Table 2: Results of Experiment 2

Platform	Algorithm	Trial 1 (D (m)/ R (rad))	Trial2	Mean
Husky	Explore Borders (This work)	189/307	230/426	210/367
Husky	Frontier Exploration	176/201	-	176/201
Kobuki	Explore Borders (This work)	290/743	312/722	301/733
Kobuki	RRT	318/652	-	318/652

Repeating the Experiments

The complete code required to build the package and repeat the experiments have been made available at https://github.com/engr-sas/explore_borders. To run the algorithm, execute the following steps:

Install ROS, Kinetic Distribution or later.

i. Download the source code from above web address.

ii. Copy to workspace and build the package.

iii. The package contains launch files and worlds that may be run out of the box. These examples are in path: border_search/launch/demo. The examples launch the experiments presented in aboveand therefore require other external packages. Table 5.3 is a list the required packages with web address to instructions on installing or cloning their respective repositories. This list does not include dependencies like move_base or ros_msgs.

Table 3: Resources Needed for Repeating Experiment

Package	Web address
Husky Robot	http://wiki.ros.org/Robots/Husky
Kobuki Robot	http://wiki.ros.org/kobuki
Frontier exploration	http://wiki.ros.org/frontier_exploration
RRT Exploration	http://wiki.ros.org/rrt_exploration http://wiki.ros.org/rrt_exploration/Tutorials

For a less restrained run of the algorithm ensure the following:

i. A mobile base equipped with a properly configured “move_base” node.

ii. A SLAM node for map creation.

iii. Properly configured parameters for border_search. See example launch files.

Discussion

The algorithm presented had a performance similar to “RRT exploration” but underperformed compared to “frontier exploration”. This assessment is based on the results of experiments presented in this work. This is however the first iteration of a novel approach and it is expected that it can be improved. The primary intention reducing translation across distant regions of the exploration region was solved. The performance is however hindered by the wall following behavior used in exploring the immediate surroundings of the initial path when obstacles are detected. It is hereby postulated that if the initial path is retained but a frontier cells approach is used to explore the immediate surroundings it should lead to improved performance. This can also be the basis for future works.

The algorithm was specifically designed for sparsely populated environments like forests, factory floors, open workshops, etc., rather than buildings or other densely populated spaces. The tests here was thereby limited to such environments, the algorithm is expected to perform poorly in buildings. However, versions of it can be made for buildings, or tuned to operate in both environments.

Conclusion

A new algorithm was designed to solve the problem of robot exploration. The algorithm was sought to solve a major drawback of the frontier cells approach. The performance of the algorithm was measured by comparing it with two other exploration algorithms. The performance metric used was the total distance and rotation undertaken by the robot in completing the exploration of a defined area.

The algorithm presented had a performance similar to “RRT exploration” but underperformed compared to “frontier exploration”. This assessment is based on the results of experiments presented in this work. The new algorithm presented was shown to be a feasible solution for certain exploration problems. It is especially promising in sparsely populated environments, or open fields with small objects.

A variation of the algorithm to be considered might use a local frontier cells approach instead of a wall following behavior.

Acknowledgment

This work used data that was generated during a Master of Science thesis at Beijing Institute of Technology BIT. However, the work presented here has never been published. The masters and by extension this work was sponsored by a joint scholarship from the Nigerian government (Bi-lateral scholarship scheme) and the Chinese Scholarship Council (CSC).

References

[1] B. Kuipers and Y. T. Byun, "A qualitative approach to robot exploration and map-learning," Robotics and Autonomouns Systems, pp. 47-63, 1987.

[2] U. Jain, R. Tiwari and W. W. Godfrey, "Comparative study of frontier based exploration methods," in Conference on Information and Communication Technology (CICT), 2017.

[3] I. Noreen, A. Khan and Z. Habib, "Optimal Path Planning using RRT* based," International Journal of Advanced , 2016.

[4] B. Yamauchi, "Frontier-Based Exploration Using Multiple Robots," in Proccedings of the nternational Conference on Autonomous Agents, 1998.

[5] Y. Wang, A. Liang and H. Guan, "Frontier-based multi-robot map exploration using Particle Swarm Optimization," in IEEE Symposium on Swarm Intelligence, 2011.

[6] M. Faria, I. Maza and A. Viguria, "Applying Frontier Cells Based Exploration and Lazy Theta* Path Planning over Single Grid-Based World Representation for Autonomous Inspection of Large 3D Structures with a," Journal of Intelligent & Robotic Systems, pp. 113-133, 2019.

[7] M. Juliá, A. Gil and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," Autonomous Robots, pp. 427-444, 2012.

[8] D. H. W. B. a. G. G. C. Stachniss, "On actively closing loops in grid-based FastSLAM," Advanced Robotics, pp. 1059-1079, 2005.

[9] A. A. Makarenko, S. B. Williams, F. Bourgault and H. F. Durrant-Whyte, "An experiment in integrated exploration," in IEEE/RSJ International Conference on Intelligent Robots and System, 2002.

[10] D. P. Ström, I. Bogoslavskyi and C. Stachniss, "Robust exploration and homing for autonomous robots," Robotics and Autonomous Systems, pp. 125-135, 2017.

[11] T. Kollar and N. Roy, "Trajectory Optimization using Reinforcement Learning for Map Exploration," The International Journal of Robotics Research, pp. 175-196, 2008.

[12] R. Zlot, A. Stentz, M. B. Dias and S. Thayer, ""“Multi-robot exploration controlled by a," in Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), Washington, 2002.

[13] A. KHAN, I. NOREEN and Z. HABIB, "On Complete Coverage Path Planning," Journal of Information Science and Engineering, pp. 101-121, 2017.

[14] GAZEBO, "Information on Gazebo," 09 may 2019. [Online]. Available: http://http://gazebosim.org.

[15] Kobuki, "Kobuki Robot Characteristics," 09 05 2019. [Online]. Available: http://http://kobuki.yujinrobot.com.

[16] ClearPath, "HUSKY ROBOT," 09 05 2019. [Online]. Available: https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot.

[17] H. Umari and S. Mukhopadhyay, ""“Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in IEEE/RSJ International Conference on ntelligent Robots and Systems (IROS), Vancouver, 2017.