Implementation of Frontier-Based Exploration Algorithm for an Autonomous Robot

Erkan Uslu, Furkan Çakmak, Muhammet Balcılar, Attila Akıncı, M. Fatih Amasyalı, Sırma Yavuz
Computer Engineering Department
Yıldız Technical University
İstanbul, Turkey
{erkan, furkan, muhammet, mfatih, sirma}@ce.yildiz.edu.tr, akinci.attila@gmail.com

***Abstract*— Exploration is defined as the selection of target points that yield the biggest contribution to a specific gain function at an initially unknown environment. Exploration for autonomous mobile robots is closely related to mapping, navigation, localization and obstacle avoidance. In this study an autonomous frontier-based exploration strategy is implemented. Frontiers are defined as the border points that are calculated throughout the mapping and navigation stage between known and unknown areas. Frontier-based exploration implementation is compatible with the Robot Operating System (ROS). Also in this study, real robot platform is utilized for testing and the effect of different frontier target assignment approaches are comparatively analyzed by means of total path length and thereby total exploration time.**

Keywords— exploration, frontier-based, real robot platform.

I. INTRODUCTION

One of the most important key points of an autonomous robot is its ability to construct spatial models and maps about their environments. Autonomous robots need good exploration strategy to effectively perform the incremental mapping function for partially known areas. There are so many exploration algorithm available under different headings depending on the structure of maps (grid, point, line-based, metric, and topological) and goals of the target selection criteria (costs, earnings).

In [1], an exploration of the robot to move along concentric circles is proposed. In [2], proposal is that the robot should select randomly safe destination to make progress with the exploration, and visit all unknown areas. As well as in the literature, the next step to find the best algorithms for autonomous robot, greedy based methods has been proposed. In this approach, the best next step (frontier) will be found as a result of processing alternatives with an evaluation function. In [3], alternative targets are evaluated combining, unexplored region of space that can be visible from the target point and the distance to a target point, with cost/benefit functions. In this method, unexplored areas that can be seen from the target point are defined as the unexplored area ratio of a certain radius around the target point. In [4], the relative entropy-based alternative target evaluation was offered. When evaluating alternative targets, the contribution of all the points to be seen from the target point, the contribution of the points to be seen for the first time from the target point, the contribution of previously discovered points as seen from the target point and the effect of transportation costs to the destination, are taken into account.

In this work, autonomous exploration application has been implemented using frontier targets method. In section II required methods for a whole autonomous robot apart from exploration are defined. In section III, frontier targets method and steps have been described. In section IV real robot platform and the ROS framework is detailed. Section V is devoted to the experimental design and results. The general conclusions and the future work are given in section VI.

II. METHODS

The complementary methods that are required for a whole autonomous robot implementation are given and defined below.

A. Laser Scan Matcher (LSM)

LSM is a form of visual odometry that is calculated by the help of matching consecutive laser scans. The result from this visual odometry is fed into the mapping algorithm for faster convergence. The LSM algorithm used in this study [5] is based on point-to-line metric iterative closest point (PLICP) which is an enhanced version of iterative closest point (ICP).

ICP algorithm is an iterative method that calculates the required trans-rotation (rotation $R(\theta)$, translation t) which registers $\{p_i\}$ point set to a reference S^{ref} surface. For a given point set $\{p_i\}$, registering trans-rotation q can be given in (1).

$$p \oplus q = p \oplus (t, \theta) \triangleq R(\theta)p + t \tag{1}$$

ICP tries to find the optimum q which minimizes the distance between, transformed p_i and Euclidean projection of the transformed p_i on S^{ref}. ICP minimization constraint function is given in (2). Here, $\boldsymbol{\Pi}\{S^{ref}, p\}$ is the Euclidean projection on S^{ref}.

$$\min_q \sum_i \left\| p_i \oplus q - \boldsymbol{\Pi}\{S^{ref}, p_i \oplus q\} \right\|^2 \tag{2}$$

This research has been supported by Turkish Scientific and Technical Research Council (TUBITAK), EEEAG-113E212 and Yıldız Technical University Scientific Research Projects Coordination Department (Project Numbers: 2015-04-01-GEP01 and 2015-04-01-KAP01).

There is no close form solution for (2). Therefore, an iterative constraint function based on an initial q_0 trans-rotation is given in (3).

$$\min_{q_{k+1}} \sum_i \left\| p_i \oplus q_{k+1} - \mathbf{\Pi}\{S^{ref}, p_i \oplus q_k\} \right\|^2 \tag{3}$$

For different definitions of $\mathbf{\Pi}\{S^{ref},\cdot\}$ various ICP approaches can be given. PLICP algorithm defines the closest line distance to a given point in closed form. While point to point registering approaches converge linearly, PLICP converges quadraticly. PLICP constraint function is given in (4) where n_i^T is the transpose of the normal of closest line on the reference line to a given point.

$$\min_{q_{k+1}} \sum_i (n_i^T [p_i \oplus q_{k+1} - \mathbf{\Pi}\{S^{ref}, p_i \oplus q_k\}])^2 \tag{4}$$

PLICP algorithm is defined below with y_{t-1} reference laser scan, y_t second laser scan, q_0 initial trans-rotation value, i second laser scan's indexes, j reference laser scan indexes, and k iteration step.

Algorithm PLICP
Input: y_{t-1}, y_t, q_0
$S^{ref} \leftarrow$ partial line segments compose of y_{t-1} $k \leftarrow 0$ repeat $p_i^w \leftarrow p_i \oplus q_k$ $j_1^i, j_2^i \leftarrow$closest two points to p_i^w $\left(j_1^i, j_2^i \in y_{t-1}\right)$ $C_k \leftarrow$ all triples $\langle i, j_1^i, j_2^i \rangle$ clean up outliers from C_k $J(q_{k+1}, C_k) \leftarrow \sum_i \left(n_i^T \left[R(\theta_{k+1}) p_i + t_{k+1} - p_{j_1^i} \right] \right)^2$ find the minimizing $q_{(k+1)}$ for J $k \leftarrow k$+1 until (max _iteration_count) or (converge)
Output: q

B. gMapping

Simultaneous localization and Mapping (SLAM) task is carried out by gMapping [6] algorithm, utilizing only laser range sensor. With the help of LSM, gMapping can select an optimum initial start point.

gMapping is based on Rao-Blackwellized Particle Filters (RBPF) which uses grid based mapping with multiple particles. Each particle has its own belief on robots previous positions and constructs its own map. Each new laser scan updates the particle's belief. Particle beliefs consist of robot position and orientation. gMapping has also its own laser scan match for an optimization step.

gMapping algorithms is given below:

1) Measurement: Obtaining new laser scan.
2) Scan Matching: Previous and current scans are matched.
3) Sampling: Best representing new particles $\{x_t^{(i)}\}$ are calculated from proposed distribution $\pi(x_t \mid z_{1:t}, u_{0:t})$ with the help of previous particles $\{x_{t-1}^{(i)}\}$.
4) Weighting: Each partices is given a weight $w^{(i)}$ based on (5).

$$w^{(i)} = \frac{p\left(x_t^{(i)} \mid z_{1:t}, u_{0:t}\right)}{\pi(x_t^{(i)} \mid z_{1:t}, u_{0:t})} \tag{5}$$

5) Resampling: Particles with lower weights are replaced by better representing particles with higher weights.
6) Mapping: Map $m_t^{(i)}$ is calculated based on each position sample $x_t^{(i)}$ and all observations $p\left(m_t^{(i)} \mid x_{1:t}^{(i)}, z_{1:t}\right)$.

gMapping method uses occupancy grid metric model type maps. Occupancy grid enables fusion of different sensor sources and can run smoothly in high resolution grids. However in large areas as the number of grids increase, calculation cost of gMapping increases dramatically [7]. Occupancy of a grid can be defined as the ratio of times that a grid is seen as full to total times of a grid is seen by the sensor. Unexplored areas are initialized with 0.5 as the occupancy value.

$$p(x, y) = \frac{\#occupied}{\#occupied + \#empty} \tag{6}$$

C. Navigation Stack

Navigation Stack is supplied by ROS framework. Navigation stack basically is a controller producing meaningful velocity commands that enables a robot to reach a given goal, avoiding obstacles, and strictly following the calculated path with the help of robot position (odometry), and sensor information. Navigation Stack is formed as a finite state machine.

Robot size is also given as an input to Navigation Stack. So, the Navigation Stack can calculate traversable paths avoiding stuck situations. A robot to be compatible with Navigation Stack should provide following configurations [8]:

1) Robot should accept linear velocity commands and also angular velocity modeled as (x, y, θ).
2) There should be a laser measurement sensor for localization and pose estimation.
3) Robot footprint should be provided as a configuration file in Navigation Stack.

In Fig. 1 every possible ROS framework configuration with the Navigation Stack is given.

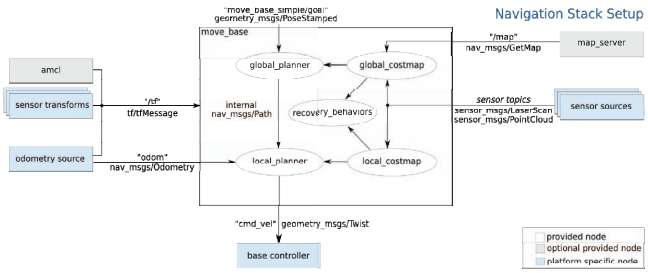


Fig. 1. ROS navigation stack configuration.

III. FRONTIER-BASED EXPLORATION

Frontier based exploration was first introduced by [9]. Frontiers are defined as the border points that happen to lie between explored and unexplored areas which are calculated at the time of mapping and navigation. Navigating to a frontier point enables exploration of unseen areas and adding information to the map. Therefore sequentially advancing to some next frontier defines the frontier based exploration.

Let all points that are traversable by a robot in a known map, defined as reachable points. Each reachable point can be defined as neighbor and adjacent, as it is possible to define at least one path that connects the robot start point with any reachable point. In partially known maps it is obvious that previously defined paths should pass through the partially known areas. It should also be noted that any path that connects points from partially known area to unexplored areas should pass through a frontier point. Following a path up to several frontiers that connects points from known area to unexplored area each path can be achieved. A robot then can map the entire unexplored area accepting perfect motor control and perfect sensor information. Frontier based exploration guaranties fully mapping of an unknown environment in finite time but with decreasing amount of marginal information as the explored area expands. However in practice noisy sensor, motor and driving should be taken into account.

while (!map_covered & !no_frontier_with_enough_size)
  $M \leftarrow$ Current_map
  $R \leftarrow$ Robot_position
  $F \leftarrow$ Find_frontier_points (M)
  $F_c \leftarrow$ Distances_based_connected_component(F)
  foreach F_c
    $F_{c_g} \leftarrow$ Find_cluster_center (F_c)
  $F_{target} \leftarrow \text{argmin}_g \left\{\text{Traversible\_distance}\left(F_{c_g}, R\right)\right\}$

Candidate frontier points are explored area points with unexplored area point neighbors. In practice it is not feasible to take into account each candidate frontier point. Therefore a point is selected as a candidate frontier point if and only if any 5×5 neighborhood with an explored area point at its center has predefined number of explored and unexplored neighbors but no obstacle points.

Frontier points are then clustered with distance based connected component method. Cluster centers are calculated for the clusters that has higher number of points than a predefined cluster size threshold. A cluster center that maximizes some selection criteria is assigned as the current exploration goal.

Exploration goals cannot be always guaranteed to be as reachable before the robot starts its navigation. Such example is given in Fig. 2 where an obstacle shadow intersects to form a triangular frontier cluster (indicated with yellow lines) and the cluster center (indicated with a red dot) is calculated to be on the obstacle (indicated in blue), yet the robot is unaware of the goal being unreachable throughout its given navigation path. Such situations should be taken into account properly at navigation and exploration levels.

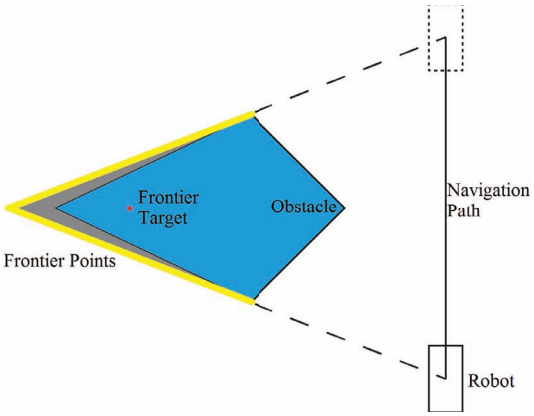
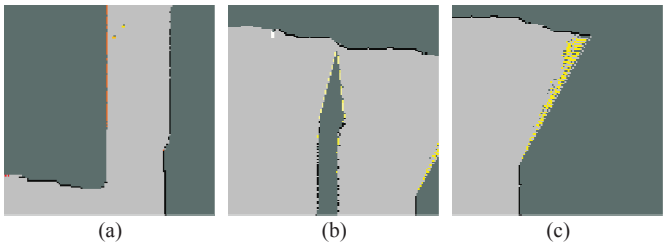


Fig. 2. A state where frontier target is calculated on an obstacle.

Fig. 3 summarizes different types of frontier clusters that are encountered. In Fig. 3(a) frontier cluster that forms when a narrow L type passage is seen from the one end is given. The cluster seen in Fig. 3(b) is formed when sensor shadow intersects behind the obstacle as it is seen from left and right sides. Frontier cluster given is Fig. 3(c) can be formed when an open area is seen from distant.



(a) (b) (c)

Fig. 3. Different types of frontier clusters, (a) line like, (b) triangular, (c) scattered.

IV. SYSTEM OVERVIEW

A. Mobile Robot Platform Hardware

The mobile robot platform used during the tests is constructed as part of a research project aiming to develop exploration, mapping and victim detection algorithms for autonomous search and rescue robots.

Current capability of the wheeled robot platform is somehow limited but sufficient enough for initial development and testing of the algorithms on partially irregular terrains. The drawings and the picture of the four wheeled, differential drive robot platform is given in Fig. 4.

Fig. 4. Real robot platform.

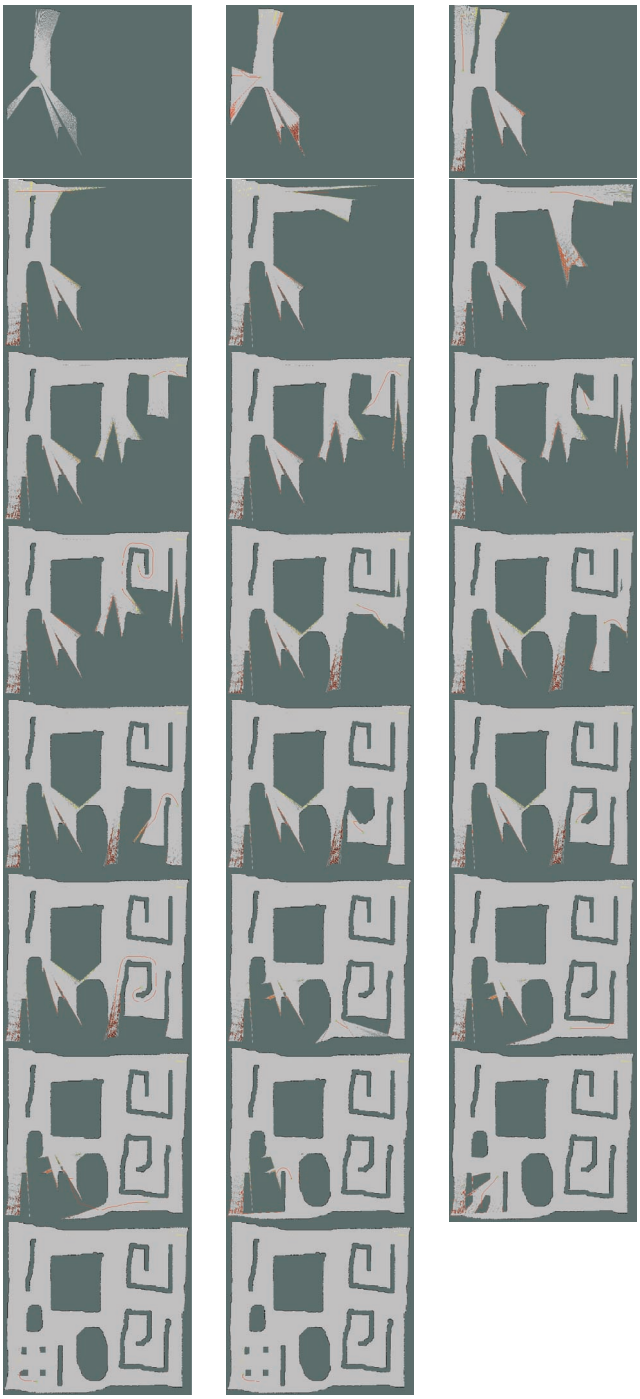


Fig. 5. Sim-A exploration results based on nearest frontiers.

The robot is equipped with different sensors including an RGB-D camera, one LRFs and an inertia measurement unit (IMU) that may be used for exploration and mapping purposes. Additionally, thermal camera, microphone and carbon dioxide sensors are used to detect the victims and to determine their states.

However, for the implementation of frontier based exploration here, LRF is the only required sensor to produce 2D map of the environment. In this study, the inertia measurement unit IMU is also utilized only to control and stabilize the LRF. LRF is fixed to a base on top of a pan/tilt unit and the angles of the unit are controlled to be equal to the negative of the angles measured by the IMU. As a result, the LRF direction is stabilized to always be level and pointing in the same direction. This eliminates the requirement of identifying and removing any noisy or invalid range scans within the algorithm.

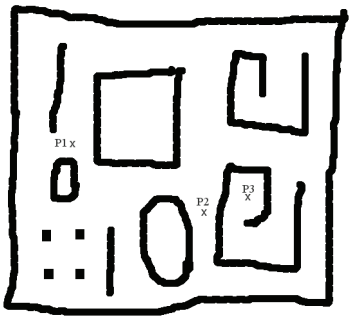


Fig. 6. Sim-A Map with different starting points.

B. Robot Operating System (ROS)

Robot Operating System (ROS) [10] is a widely used framework by the robotics community. ROS is an open source software released under the terms of the BSD (Berkeley Software Distribution) license and promotes code reutilization for the robotics developers. Currently a number of research institutions develop their projects in ROS by adding hardware and sharing their code samples. Codes written for ROS can be used with other robot software frameworks and implementation in any modern programming language is supported. The ROS framework is used in this study for both hardware and software implementations.

V. Experimental Setup and Results

The experimental results were obtained in two different ways. The first one is obtained in simulated environment and the second one is real environment (Simulated map of the real environment is also used).

The autonomous exploration algorithm is implemented using ROS framework. "stage_ros", "rviz", "gmapping", "move_base" packages of ROS are used. As a local planner, dynamic window approach (DWA) and as a global planner, “navfn” package add-ons are used with “move_base” package. In DWA method, robot control space is discretized and each control pair (v_x: linear velocity, ω_z: angular velocity) is evaluated/simulated for a short period of trajectories. The highest score route in terms of speed or the shortest route to obstacles, targets, the main routes, which will impact the barriers, are selected and applied. Destination route is calculated by Dijkstra's algorithm within “navfn” package which is used as global planner.

In the determination of the frontier points, evaluation is made with a 5×5 size local window. If there are 10 empty and 8 unexplored grids around local window, this window’s empty center is selected as frontier point. The cutoff value used in, clustering the frontier points, is selected as 25 cm, for the

distance based implementation of the connected component analysis. Neighborhood-based distance radius of 3 grids is used for resolution.

Initially a clearing rotation is executed and then in a partially explored map consecutively selecting frontier targets the exploration can be carried out with respect to a target selection criterion.

A. Simulated Environments

Two different simulated environments are used. The first one (Sim-A), a map simulation environment, which has U shaped, helical, rectangular and circular rooms. This map includes various geometric barriers and spiral rooms. Some passages between rooms are too narrow and some of them are wide. The second simulation environment (Sim-B) is obtained from the real environment with some abstractions.

The area of Sim-A is approximately 50×50 m^2 and its grid resolution is given as 0.05 m. Sim-A environment is given in Fig. 6. Stage [11] simulation environment was used for the experimental results. A simulated laser distance sensor, which has 270-degree viewing angle and 30 m distance measurement capacity, is used.

In Fig. 5, each step of the nearest frontier exploration is represented visually providing calculated frontier points, frontier clusters, cluster centers, robot position, and calculated paths.

In Fig. 5 and other exploration figures different color codes different frontier clusters. Current calculated global path is drawn in solid red line. Robot footprint is given in green rectangle. Unexplored areas are colored in gray whereas explored/cleared areas are given in white. Obstacles are colored in black.

Obstacle avoidance is carried out with the help of local and global cost maps. Global cost map is produced with the map constructed by the mapping algorithm whereas local cost map is constructed by the laser sensor, ranging within a predefined radius.

Experimentally, three different frontier target selection approaches, namely random, the biggest, the nearest, are applied for comparison of total paths and total time to finish the exploration. Three different starting points that are given in Fig. 6, are used as initial exploration points for further comparison.

TABLE I. NEAREST, BIGGEST AND RANDOM TOTAL PATHS AND TOTAL TIMES FOR DIFFERENT STARTING POINTS.

Starting Points		Frontier Target Selection Approaches		
		Nearest	Biggest	Random
P1	Time (sec)	1479	2086	2142
	Path (m)	331	500	449
P2	Time (sec)	2595	3089	2202
	Path (m)	577	686	469
P3	Time (sec)	1610	1492	1581
	Path (m)	345	340	370

In Table I, total paths and total times that are calculated are listed based on different starting points and according target selection approaches.

According to Table I, nearest frontier target selection approach yields better results compare to biggest and random results. The number of times, the robot comes across with the situations that is defined in Fig. 2 can be given as follows: (Random) > (Nearest) > (Biggest).

It is expected that nearest frontier selection approach should minimize the revisits of a partially explored rooms/halls. However, even with nearest frontier target selection, substantial amounts of revisits occurred. So, it is concluded that revisit minimization should be handled along with nearest frontier target selection.

Sim-B environment is the abstraction of the real environment that is simulated as 100×55.5 m^2 area in 2.5 cm grid resolution. Sim-B contains a wide corridor, empty rooms, and rooms with student laboratory desks and chairs. The abstracted map of Sim-B that is used for both simulation and real environment is given in Fig. 7.

Simulation steps of the frontier based exploration using the nearest frontier target selection for Sim-B after a more that one hour run is given in Fig. 8.

B. Real Environment

Real robot platform is used for autonomous frontier-based exploration in real environment. The results for two mini runs are presented with Fig. 9 and Fig. 10 visually. In Fig. 9 and 10 RGB-D camera images are also visible. The trajectory of the robot is given in blue color. The real environment is mostly constructed by office furniture and some empty rooms.

Real environment results are mostly compatible with the results of abstract map of the environment used in Sim-B, however since overseeing behind the obstacles that are glass or seeing through not fully surrounding obstacles some unreachable frontier clusters are formed in the real environment. In such situations to prevent consecutive unreachable frontier target point selection, aborted target points and its surrounding is kept in a list of unreachable frontiers. An example for such a situation is given at the last image of Fig. 10.

Fig. 7. Sim-B and real environment floor plan.

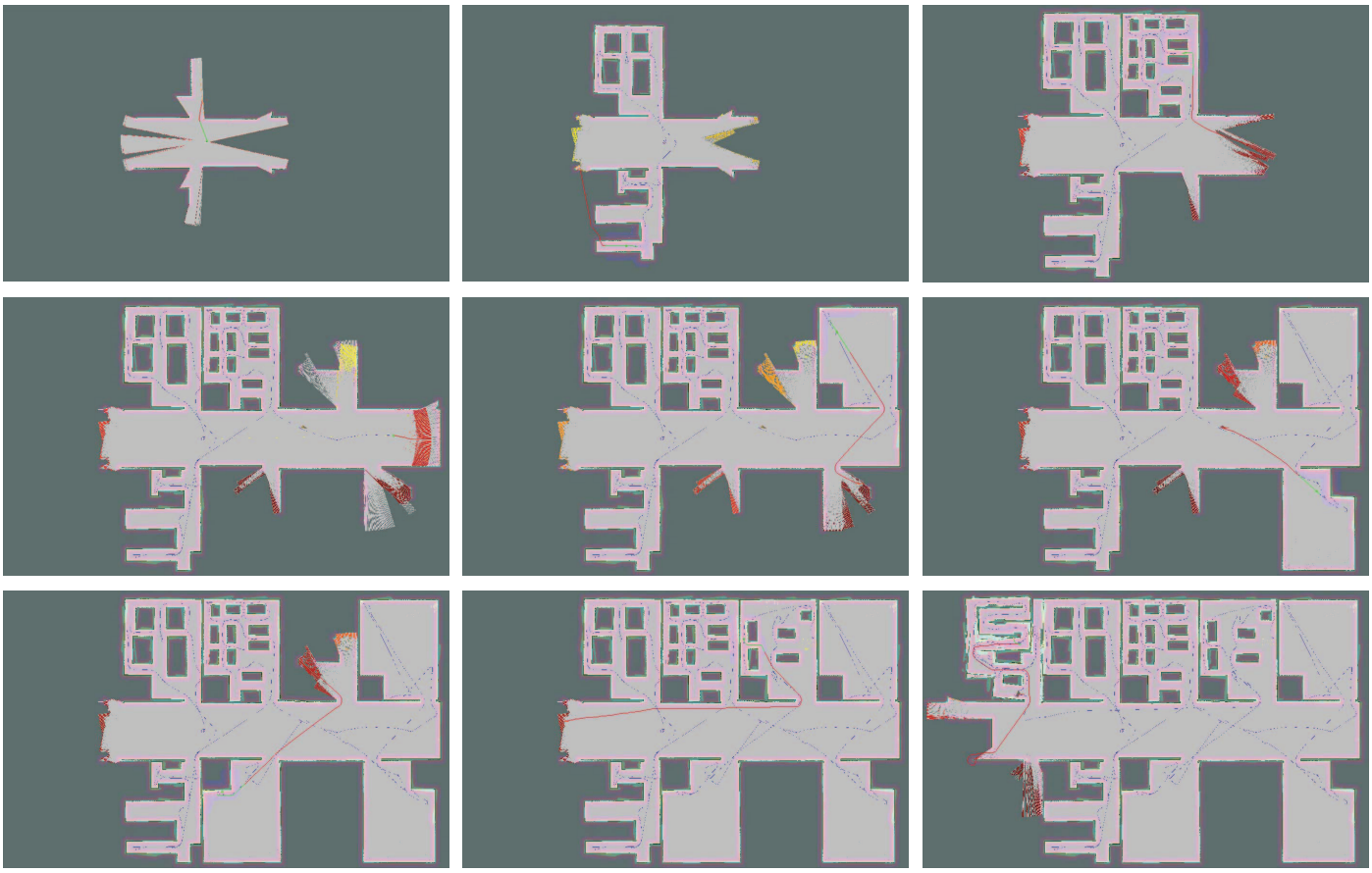


Fig. 8. Sim-B exploration steps.

Fig. 9. Real environmen run 1 exploration steps.

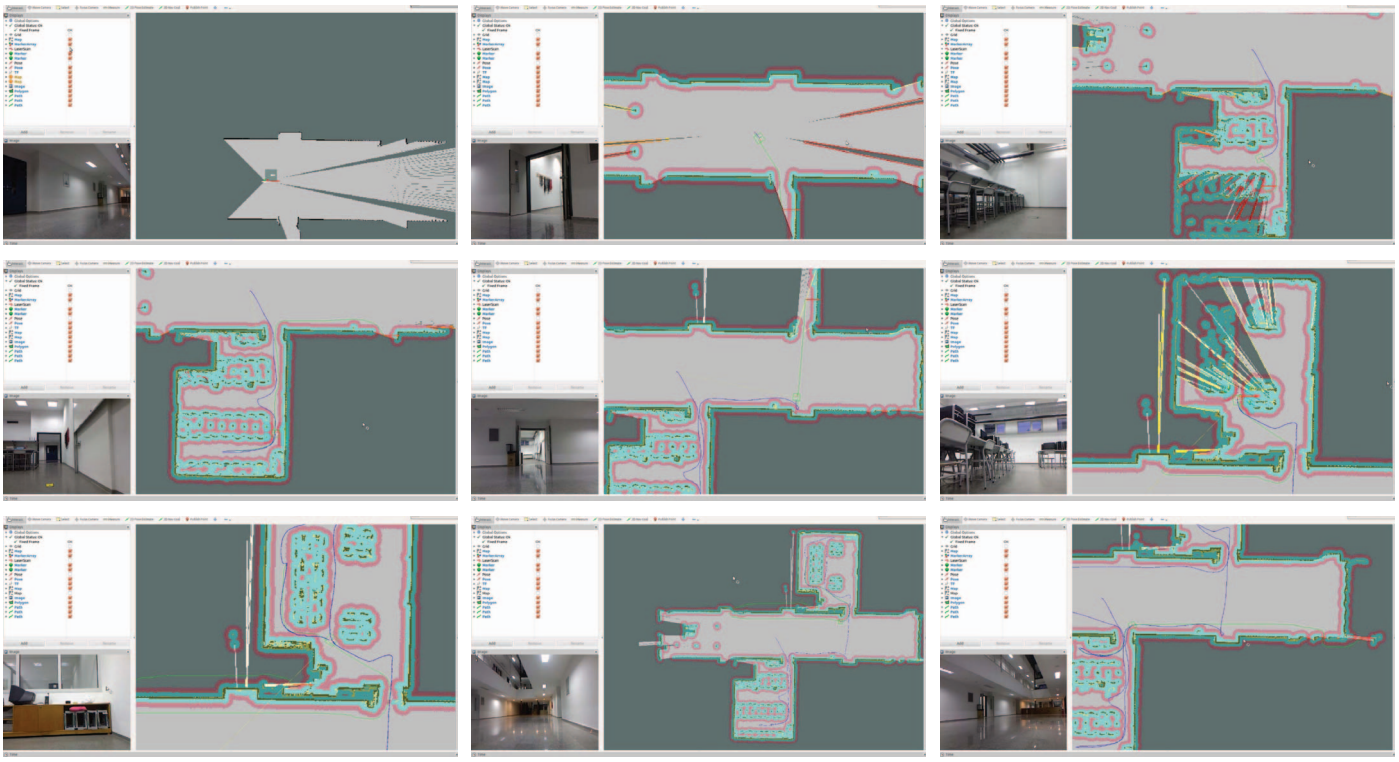


Fig. 10. Real environmen run 2 exploration steps.

VI. Conclusion

In this study, ROS compatible frontier based exploration is implemented and various simulation and real environment results are presented. For the simulation, Stage environment is utilized. Based on the results from different scenarios, applied in the simulation environment optimum parameters are calculated and used in real environment.

In future works, to be able to minimize the revisit to partially explored areas, segmentation based room extraction is considered. As well as, frontier targets can become explored during the navigation. So, moving the robot to the same frontier target can be considered as loss of time. Also, as a future work, it is considered to recalculate new frontier targets when a previous target becomes explored.

It is also concluded from the study that accompanying different frontier target selection approaches, biggest, nearest, random, at some special occasions can decrease the total time of exploration and the total distance of travel.

References

[1] R. Sim and G. Dudek, "Effective Exploration Strategies for the Construction of Visual Maps," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 2003.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] L. Freda and G. Oriolo, "Frontier-Based Probabilistic Strategies for Sensor-Based Exploration," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005. K. Elissa, “Title of paper if known,” unpublished.

[3] H. H. Gonzales-Banos and J.-C. Latombe, "Navigation strategies for Exploring Indoor Environments," *The International Journal of Robotics Research,* vol. 21, no. 10-11, pp. 829--848, 2002.

[4] F. Amigoni and V. Caglioti, "An information-based exploration strategy for environment mapping with mobile robots," *Robotics and Autonomous Systems,* vol. 58, no. 5, pp. 684-699, 2010.

[5] A. Censi, "An ICP variant using a point-to-line metric," *Robotics and Automation, 2008. ICRA 2008. IEEE Int. Conf. on.*, pp.19,25, 19-23 May 2008.

[6] G. Grisetti; C. Stachniss.; W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *Robotics, IEEE Transactions on* , 23(1):34-46, 2007.

[7] H. Karaoğuz, Ö. Erkent, H. Bayram and H. I. Bozma, "Tek Robottan Çoklu Robotlara Ortam Haritalama," *EMO Bilimsel Dergi,* vol. 2, no. 4, pp. 105-118, 2012.

[8] A. Martinez, E. Fernandez. Learning ROS for Robotics Programming, Livery Place 35 Livery Street Birmingham B3 2PB, UK: Packt Publishing Ltd., 2013.

[9] B. Yamauchi, "A Frontier-Based Approach for Autonomous Exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, 1997.

[10] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng,. "Ros: an open-source robot operating system. ", In ICRA Workshop on Open Source Software, 2009.

[11] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence,* vol. 2, no. 2-4, pp. 189-208, 2008.