Active SLAM-Based Algorithm for Autonomous Exploration with Mobile Robot

Darko Trivun, Edin Šalaka, Dinko Osmanković, Jasmin Velagić, Nedim Osmić
Faculty of Electrical Engineering
Department of Automatic Control and Electronics
University of Sarajevo
71000 Sarajevo, Bosnia and Herzegovina
Email: {dtrivun, esalaka, dosmankovic, jvelagic, nosmic}@etf.unsa.ba

***Abstract*—In this paper, we present an algorithm for fully autonomous exploration and mapping of an unknown indoor robot environment. This algorithm is based on the active SLAM (simultaneous localization and mapping) approach. The mobile robot equipped with laser sensor builds a map of an environment, while keeping track of its current location. Autonomy is introduced to this system by automatically setting goal points so that either previously unknown space is mapped, or known landmarks are revisited in order to increase map accuracy. Final aim is to maximize both map coverage and accuracy. The proposed procedure is experimentally verified on Pioneer 3-DX mobile robot in real environment, using ROS framework for implementation.**

I. INTRODUCTION

SLAM is a popular and widely used technique for mapping an unknown environment that is either static or changes very slowly (low movement in area while certain landmarks are still static) [1]-[3]. It is usually combined with human activity and supervision, since we have a completely unknown environment. There are not many research works going on with goal of designing an off-line algorithm that would cover the whole area efficiently [4]-[6].

Recent works in this area are focused on designing an on-line algorithm that estimates current position and previous results and sets goal point in order to maximize the map coverage and accuracy [5]-[7]. Some of recently proposed techniques include methods based on Kullback-Leibler divergence for evaluating the SLAM posterior approximations [7]; hierarchical Bayesian approach to determine what area should be visited next [6]; model predictive control and using an attractor to incorporate long term goals [8]; actively closing loops that are made during the exploration [5].

Algorithm presented in this paper takes previous mapping results into account, and combines them with costmaps that are used for autonomous navigation. In such way, we can determine "cheapest" goal point that will also increase coverage and accuracy of our map. Goal points are chosen by combining multiple criteria. First, we have to make distinction between the areas that lies outside the range of the sensor, and the areas that are close enough to be mapped, but are hidden by an obstacle. Then we weight possible actions and pick the most favorable. Once goal points are set, we need to build efficient path from current position, and also to provide a moving the robot safely without colliding with static obstacles (e. g. wall) or dynamic ones (e. g. human or another robot).

In our paper, the control system is used for the robot navigation. Deliberative layer is based on A* search, while reactive layer uses dynamic window approach. Some alternate algorithms that could have been used for deliberative search are Dijkstra's search, which is equivalent to running A* without heuristic functions [9], or some algorithm from D* family, which are specific because they are searching from the goal to the start [10] and [11]. Also, other popular approaches for reactive layer are the ones using potential fields [12] and [13]. Such methods are based on applying attractive force to the goal point, and repulsive force to the obstacles, so we can form the resulting force field which will guide the vehicle to the goal point. Finally, an environment is mapped using one variant of SLAM algorithm, FastSLAM. FastSLAM relies on particle filter, specifically Rao – Blackwellized filter. Other SLAM algorithms include EKF SLAM class of algorithms, which are based on extended Kalman filter. Their main disadvantage is fact that they deal badly with uncertainty due to Gaussian noise assumption [14]. The whole proposed system is verified in simulated environment using ROS and Stage simulator, and then on the real robot, specifically Pioneer 3-DX, while also using ROS as a framework.

This paper is organized as follows. Section II describes used physical system, along with all the techniques used in this paper. In Section III, detailed explanation of proposed algorithm is given. Section IV presents the results of the proposed algorithm tested in both simulation and real world environments. Finally, the paper is concluded by Section V.

II. USED SYSTEM AND TECHNIQUES

A. Physical System

Robot that was used for verification of the proposed algorithm is Pioneer 3-DX (P3-DX). P3-DX is differential drive robot. It is equipped with a complete sonar ring, wheel encoders and embedded motion controllers that are responsible for low level control of velocity and direction of motors [15]. Also, the SICK LMS220 laser range finder is attached to the robot base (Fig. 1). LMS stands for laser measurement system, and it is primarily used for monitoring areas, position determination, vehicle guidance, and collision control [16]-[18]. In this paper, laser is used as a distance sensor, which can detect obstacles, so we would be able to map the area around the robot using its readings. LMS220 laser has range limit of 8 meters.

In this paper we used Robot Operating System (ROS) as

978-1-4799-7800-7/15/$31.00 ©2015 IEEE

Fig. 1. P3-DX mobile robot with SICK laser range scanner

our framework. ROS is an open-source framework for writing robot software [19]. It is primarily used as interface between the controller (PC) and the robot itself, but the simulators and models of certain robots are also available.

B. SLAM Algorithm and Particle Filter

Particle filter is a method based on estimating the posterior density of the state-space. Estimation itself is theoretically based on Bayesian recursion equations [20]. Posterior density is estimated by set of particles. Each particle is weighted and these weights are updated through time [21]. One type of particle filter, called Rao – Blackwellized particle filter, has recently been used for solving SLAM problem [22]. This filter takes most recent observations into account as well, instead of only using position and trajectory of the mobile robot.

In this paper, an implementation of SLAM using Rao – Blackwellized particle filter called FastSLAM is used. FastSLAM, presented in [23], decomposes the SLAM problem into a robot localization problem, and a collection of landmark estimation problems that are conditioned on the robot pose estimate. Thus, posterior is always estimated over the robot path. Each particle consists of a number of Kalman filters that is equal to the number of landmark locations on the path estimate, which in the end results in one instance of Rao – Blackwellized particle filter [23].

C. Navigation Control Algorithm

In order to achieve hybrid robot control, two algorithms were used in our paper. One costmap was created for each layer. Global planner is based on A* search algorithm, while local planer uses the dynamic windows approach. Open source implementation of mentioned algorithms, and hybrid control architecture was used [24]-[27].

A* is a path finding algorithm that uses a best-first search to find the cheapest path to the goal point. Best-first search based algorithms are trying to follow cheapest node first until they reach the goal. If goal is not reached, then next node in the line is tried [28]. As A* follows a certain path, alternate paths that can be created along the way are also kept in memory, together with their costs. By doing such, the algorithm cannot get stuck in a loop, which is not the case for all best-first search algorithms. Such algorithm is called complete, since it always terminates if appropriate path exists [25].

The dynamic windows approach is a reactive algorithm, which means that all its actions are a result of a reaction to its environment, without any special planning. Thus, it is well fit to be combined with another deliberative algorithm, such as A* into a hybrid control architecture [26]. This hybrid algorithm works in real-time, and is activated when an obstacle is detected near the vehicle. What is special for dynamic windows approach is that possible solutions for avoiding an obstacle are not found by constructing a path that avoids an obstacle, but by determining what velocity and acceleration should be set in order to do that [29]. The dynamic windows that contain all possible velocities are constructed, and the velocity that optimizes the function which contains the measures for clearance and path alignment is chosen [29].

III. Proposed Active SLAM-Based Algorithm

Active SLAM is fully autonomous approach to solving SLAM problem. First step we have to make before taking any action is making distinction between frontier that lies on the edge of laser's range of effect and areas that were not mapped because they were hidden by another obstacle. We can do that easily by scanning global costmap that was used in A* search. We scan global occupancy matrix for gaps, and note whether they were orthogonal within several degrees to robot position and inside of distance sensor range at the same time. If they were not, we mark such gaps as areas hidden behind an obstacle. Such hidden areas are always taken into consideration before edge frontiers, as they are closer and usually cheaper to explore. If we have more hidden areas to consider, we pick the closest one.

If we don't have hidden areas that need to be explored, we move to the edge frontier. Our goal is to make robot orientation vector orthogonal to the edge, so that we would cover as much area as possible with as little movement as possible. Each frontier is segmented, and we plan a trajectory to each segment. After that, we monitor map and pick trajectory that would cover map segments with highest covariance. In this way we do not have to revisit specific area in order to increase accuracy, but still, accuracy is increased all the time because trajectory is planned in such fashion. Once we pick segment for exploring, the robot follows the trajectory that was already planned. If robot for any reason fails to reach the goal, we plan the new trajectory to the same spot and retry the action. If the algorithm fails again, that spot is moved to the last place on stack, so that we revisit it later, but still don't lose much time on retrying, as algorithm is not bound to be complete, and spots that are not able to be explored are always possible.

If hidden area is successfully visited, but still not mapped completely (i. e. there are still gaps in the map itself, but no better position to observe such area can be found), we move that area to last place on stack and mark it as already visited. If it is visited once again, but still not mapped correctly, we update costmap, so that algorithm does not detect that gap again. This usually happens in the corners that have lots of static obstacles that robot cannot reach safely. The pseudo code of the proposed algorithm for fully autonomous exploration and mapping of an unknown indoor environment is given in

Algorithm 1 Pseudo code of the proposed algorithm

```
Start
Initialize Stack
Run SLAM iteration, build global costmap
Procedure Fill stack:
Scan global costmap for gaps
for Each gap In global costmap do
    if gap is Not On Stack then
        if gap is Not orthogonal within 10 deg to the robot position And gap is inside sensor range then
            Set gap As Hidden
        else
            Set gap As Edge frontier
        end if
        Put gap On Stack
    end if
end for
end Procedure
```

Algorithm 2 Pseudo code of Set_current_goal procedure

```
Procedure Set current goal:
Find closes Hidden gap on Stack
if one exists then
    Set it as current goal
else
    for Each Edge frontier gap do
        Compute trajectory using A*
        Compare trajectory to the Map, calculate Sum of uncertainty
    end for
    Set Edge frontier gap with highest Sum of uncertainty ss current goal
end if
if current goal is Not set then
    end Procedure
end if
```

Algorithm 1. Pseudo code for $Move$ and $Set\_current\_goal$ procedures are given in Algorithm 2 and 3, respectively. These two procedures are not a part of the SLAM algorithm, but are employed to explore the unmapped/unknown parts of the environment.

Algorithm terminates when there are no more open frontiers, but however, it is never guaranteed to terminate, because in dynamic environment, we can always have unreachable goal because of dynamic obstacles (e. g. human or vehicle moving in front of robot fast enough, so that way is always blocked).

IV. Experimental Results

In this section, we present the results of algorithm tested in simulated and then in the real world environment. For simulation we used Stage simulator within ROS, and in the real world, we mapped a hallway that is shown in the Figs. 2, 3 and 4. Rviz package from ROS was used for visualization.

A. Test Environments

Our test environment is shown in Figs. 2, 3 and 4. It is roughly of a rectangular shape. On one side, there is a set

Algorithm 3 Pseudo code of Move procedure

```
Procedure Move:
Move to current goal, update Map
if goal is reached then
    if Map coverage around goal is Low then
        if goal is Visited then
            Update global costmap, Set corresponding gap to Hidden
        else
            Set current goal as Visited
            Move current goal to the last place On Stack
        end if
    end if
else if Not Retried then
    Set current goal as Retried
    go to Move
else
    Move current goal to the last place On Stack
end if
end Procedure
```

of stairs going up, which robot cannot take. In one corner, there are some obstacles, so that robot cannot reach the corner itself, but instead should take shots from both sides. In the other corner, there is a small passage to another hallway, but it is too narrow for robot to successfully go through. Also, at a specific time instant, dynamic obstacles in form of humans walking in front of robot are included in test case.

Before test, we expected the algorithm to explore and map the whole area, as it is not too complex, and terminate in the end, as there are no spots that should cause the robot to get stuck in a loop.

B. Obtained Indoor Maps

The robot starting position was near the place where image in Fig. 5 was taken. Initial step was to scan the area around the chair, as it was detected as area behind an obstacle. When robot started moving, one person moved into its path in order to make the passage narrower, while the other one stood near the starting position. Map (above) and costmap (below) at this stage are shown in Fig. 6, as well as trajectory that

Fig. 2. Hallway used as test case

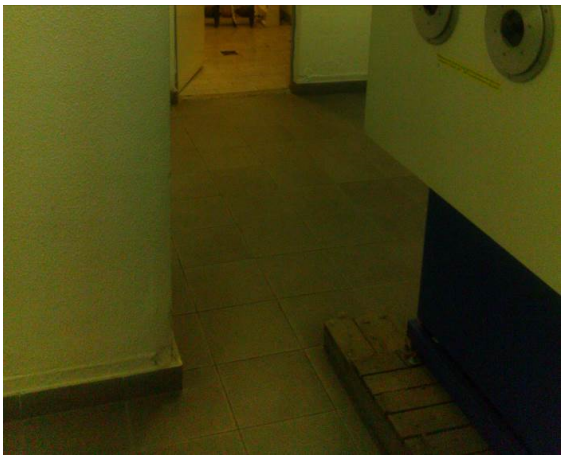

Fig. 3. End of hallway that robot was not able to pass

Fig. 4. Obstacles in corner

robot traversed on. Two persons are visible on costmap, and trajectory was planned according to their position, but still they are not present on map itself. It shows that SLAM algorithm correctly recognized dynamic obstacles and removed them from map.

As it is visible from the Fig. 6, area around the obstacle is mapped very precisely with no gaps, which makes the first test successful.

After that, next goal point was set on the frontier that was right behind the robot's starting position. Trajectory was planned so that we pass the most uncertain parts of the map. We can see that the corner was mapped successfully and also that some part of maps that were marked as uncertain in Fig. 6 are correctly marked as empty space.

Algorithm was running for some time, until we reached situation shown in Fig. 7. Robot moved in to scan the narrow corridor in the corner, but did not try to pass through it, because gap in the costmap was not large enough to consider it safe. This point was not considered again, as costmap is consulted when planning trajectory instead of real map. This is when algorithm was supposed to terminate, but we overlooked the fact that the machine shown in Fig. 9 was on the stand that was too low for our laser to detect, and a path that was supposed to be too narrow was taken into a consideration.

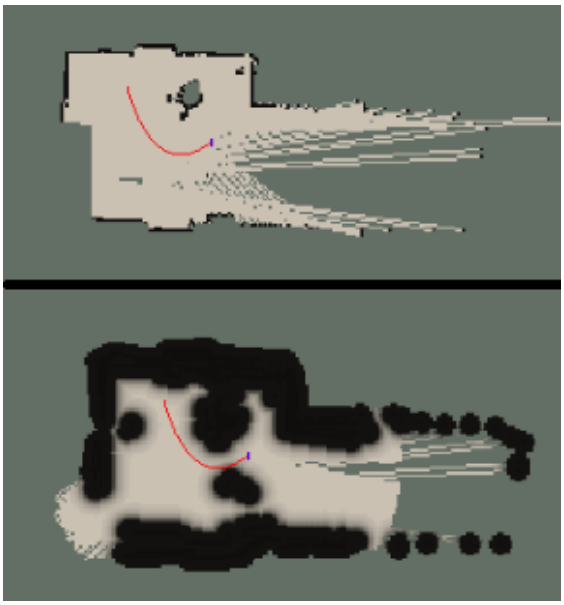

Fig. 5. Scanning the area around the obstacle from both sides

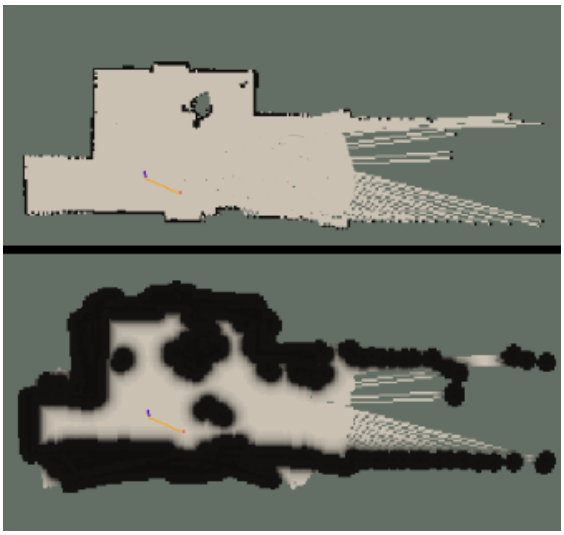

Fig. 6. Scanning the frontier while increasing accuracy of previous mappings

Robot successfully moved behind the machine, shown in Fig. 9, but the algorithm forcefully terminated when the robot hit the stand which represented an invisible obstacle. In other case, algorithm would peacefully terminate when there are no more reachable areas to be mapped in the consideration.

The final outcome of this test run is the map of this area shown in Fig. 8.

V. Conclusion

As we can see from the results, system acts as we expected of it in real world environment. The robot recognizes human movement as such, and does not mark it as an obstacle, but, nevertheless, reactive control layer is activated in order not to collide with the dynamic obstacle. In the end, precise and usable map is provided to the end user. There were slight problems in mapping tight areas and corners with lots of static

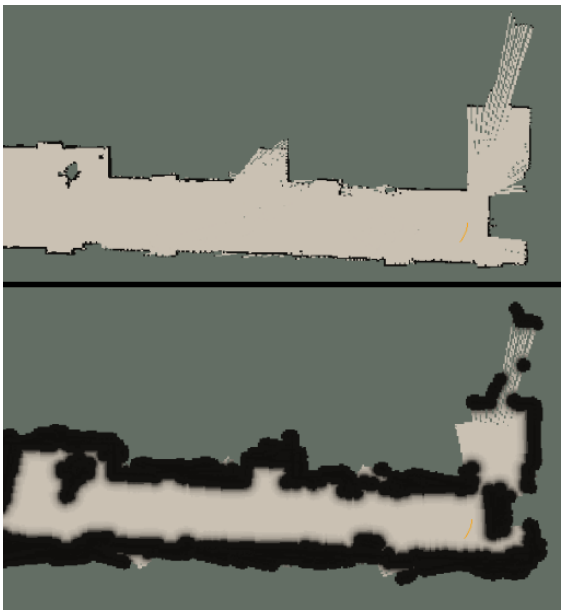

Fig. 7. Scanning the narrow pass

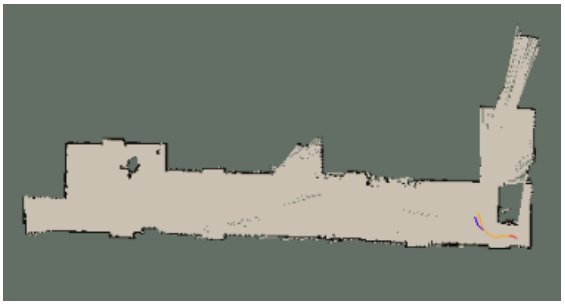

Fig. 8. Resulting map

Fig. 9. Low height obstacle that was not detected

obstacles. Also, obstacles with low height were not detected and caused problems in navigation, but these are the limitations of our robot as a physical system, and are not related to the algorithm itself. In the end, map provided has big percentage of coverage and is usable for most purposes except ones that are searching for extreme precision or coverage.

REFERENCES

[1] OpenSLAM.org. "What is SLAM?," openslam.org. [Online]. Available: http://www.openslam.org/slam.html [Accessed: Nov. 14, 2013].

[2] H. F. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): part I the essential algorithms," Robotics and Automation Magazine, vol. 13, pp. 99-110, June 2006.

[3] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in Proc. Intelligent Robots and Systems, Osaka, 1991, pp. 1442 - 1447.

[4] N. Fairfield and D. Wettergreen, Active SLAM and loop prediction with segmented map using simplified models, in Field and Service Robotics, A. Howard, K. Iagnemma, and A. Kelly, Eds. Berlin: Springer-Verlag, 2009, pp. 173-182.

[5] C. Stachniss, D. Hähnel, W. Burgard, and G. Grisetti, "On actively closing loops in grid-based FastSLAM," Advanced Robotics, vol. 19, pp. 1059–1080, Oct. 2005.

[6] D. Fox, J. Ko, K. Konolige, and B. Steward, "A hierarchical bayesian approach to the revisiting problem in mobile robot map building," in Proc. Intelligent Symposium of Robotic Research, Siena, 2003, pp. 487-506.

[7] L. Carlone, J. Du, M. K. Ng, B. Bona, and M. Indri, "Active SLAM and exploration with particle filters using Kullback-Leibler divergence," Journal of Intelligent and Robotic Systems, in press, 2014.

[8] C. Leung, H. Shoudong, and G. Dissanayake, "Active SLAM in structured environments," in Proc. IEEE International Conference on Robotics and Automation, Pasadena, 2008, pp. 1898 - 1903.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to algorithms. Introduction to algorithms, Cambridge: MIT Press, 2009.

[10] A. Stentz, "Optimal and efficient path planning for partially-known environments," in Proc. IEEE International Conference on Robotics and Automation, San Diego, 1994, pp. 3310-3317.

[11] A. Stentz, "The focussed D* algorithm for real-time replanning," in Proc. International Joint Conference on Artificial Intelligence, Nagoya, 1995, pp. 1652-1659.

[12] R. Siegwart and I. R. Nourbakhsh, Introduction to autonomous mobile robots, The MIT Press, Cambridge: MIT Press, 2011.

[13] Y. Koren and J. Borenstein "Potential field methods and their inherent limitations for mobile robot navigation," in Proc. IEEE International Conference on Robotics and Automation, Sacramento, 1991, pp. 1398-1404.

[14] D. Rodriguez-Losada, F. Matia, A. Jimenez, and R. Galan, "Consistency improvementfor SLAM-EKF for indoor environments," in Proc. IEEE International Conference on Robotics and Automation, Orlando, 2006, pp. 418-423.

[15] Adept MobileRobots. "Pioneer P3-DX," Available: http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx, 2013.

[16] LMS 200 / LMS 211 / LMS 220 / LMS 221 / LMS 291 Laser Measurement Systems Technical Description, SICK AG, Reute, Germany.

[17] M. Yang, H. Wang, K. He, and B. Zhang, "Obstacle avoidance using range data in autonomous navigation of mobile robot," in Proc. International Symposium on Test and Measurement, Xi'an, 1999, pp. 1-5.

[18] A. G. Zavodny, Change detection in LIDAR scans of urban environments, Ph.D. Dissertation, Department of Computer Science and Engineering, Graduate School of the University of Notre Dame, USA, 2012.

[19] ROS.org. "About ROS", ros.org. [Online]. Available: http://www.ros.org/about-ros/ [Accessed: Nov. 14, 2013].

[20] T. Chen, J. Morris, and E. Martin, "Particle filters for the estimation of a state space model," in Proc. 37th European Symposium of the Working Party on Computer-Aided Process Engineering, Lisbon, vol. 18, 2004, pp. 613–618.

[21] S. S. Kambhampati, K. V. Tangirala, K. R. Namuduri, and S. K. Jayaweera, "Particle filtering for traget tracking," 7th International Symposium on Wireless Personal and Multimedia Communications, Abano Terme, 2004.

[22] G. Grisseti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in Proc. IEEE International Conference on Robotics and Automation, Barcelona, 2005, pp. 667-672.

[23] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," in Proc. AAAI National Conference on Artificial Intelligence, 2002, pp. 593-598.

[24] S. Zaman, W. Slany, and G. Steinbauer, "ROS-based mapping, localization and autonomous navigation using a pioneer 3-DX robot and their relevant issues," in Proc. of the IEEE Saudi International Electronics, Communications and Photonics Conference, Riad, 2011, pp. 1-5.

[25] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige, "The office marathon: robust navigation in an indoor office environment," in Proc. International Conference on Robotics and Automation, Anchorage, 2010, pp. 300-307.

[26] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," IEEE Robotics and Automation Magazine, vol. 4, pp. 23-33, Mart 1997.

[27] GitHub, Inc. "ROS Navigation stack", github.com. [Online]. Available: https://github.com/ros-planning/navigation [Accessed: Nov. 09, 2013].

[28] L. P. Rees, A. G. Greenwood, and F. C. Siochi, "A best-first search approach for determining starting regions in simulation optimization," IIE Transactions, vol. 34, 283-295, 2002.

[29] M. Seder and I. Petrović, "Dynamic window based approach to mobile robot motion control in presence of moving obstacles," in Proc. IEEE International Conference on Robotics and Automation, Roma, 2007, pp. 1986-1991.

[30] H. Cho and C. Lan, "Hybrid shortest path algorithm for vehicle navigation," Journal of Supercomputing, vol. 49, pp. 234–247, Aug. 2009.

[31] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in Advances in Neural Processing Systems, vol. 12, pp. 1043–1049, 1999.

[32] C. Stachniss, D. Haehnel, and W. Burgard, "Exploration with active loop-closing for FastSLAM," in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, 2004, pp. 1505-1510.