Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Energy-Efficient Mobile Robot Exploration

Yongguo Mei, Yung-Hsiang Lu, C. S. George Lee, and Y. Charlie Hu
School of Electrical and Computer Engineering, Purdue University
{ymei, yunglu, csglee, ychu}@purdue.edu

***Abstract*— Mobile robots can be used in many applications, including exploration in an unknown area. Robots usually carry limited energy so energy conservation is vital. This paper presents an approach for energy-efficient robot exploration. Our approach determines the next target for the robot to visit based upon orientation information. The robot plans the path between the current position to the next target in an energy-efficient way. Our method reduces repeated coverage, a common problem for most existing utility-based target selecting methods. We conduct simulations for both random and structured environments, and compare our method with a utility-based method that chooses the middle cell from the widest opening. Results show that our method can reduce energy consumption by 42% and traveling distance by 41%.**

I. Introduction

Mobile robots can be used in many different applications, including mapping, search, rescue, reconnaissance, hazard detection, and carpet cleaning [6] [7] [11]. Exploration in an unknown area is to identify the locations of obstacles, objects, and free space by sensing the environment. Exploration is a basis for many other applications. For example, to search and rescue survivors after a disaster, robots have to explore the area to find survivors. Robots usually carry limited energy, such as batteries; thus energy conservation is an important concern for mobile robots. This paper focuses on energy-efficient robot exploration. To our knowledge, this is the first study for energy-efficient robot exploration in an environment with random or structured obstacles, such as walls.

In exploration, the robot senses the environment while moving. The robot accumulates the information from sensor data and constructs a map of the environment incrementally. At any moment, the robot needs to decide the next target to explore based upon the partial information the robot has already acquired about the environment. This is called target selection and is a fundamental problem in exploration. Target selection determines the exploring sequence of different locations, and directly affects the exploration time and the energy consumption. In general, the next target is selected from frontier cells along the border between the known area and the unknown area [15]. Many existing studies select the next target based on the utilities and costs of the frontier cells [4][13] [17]. The utility of one frontier cell is estimated based upon the size of new area that can be potentially covered at the frontier cell. The cost can be the traveling distance or energy consumption. This strategy usually can cover more new area at the beginning. However, to fully cover an area, the robot has to visit some places with small unknown areas later. This often results in repeated coverage, long exploration time and large energy consumption. Ideally, we want no repeated coverage and no crossover along the exploration path. This requires selecting targets based upon the locations of frontier cells.

This paper presents an approach for energy-efficient robot exploration. Our method is divided into two major steps as the two main contributions of this paper. (a) It is an orientation-based method for target selection. Different from the utility-based strategy, our method chooses the next target based on the robot's direction and relative location of frontier cells. Our target selection method can greatly reduce repeated coverage, and thus shorten the exploration distance and save energy. (b) It uses energy-efficient motion planning for moving from the current location to the next target. Different from many existing studies that choose the shortest route, our method estimates the energy consumption and chooses the most energy-efficient route. This requires a dedicated motion planning algorithm that is different from the algorithms planning the shortest path. We conduct extensive simulations to compare our method with one existing method that chooses the next target based upon the utility of frontier cells. Simulation results show that our method is effective in reducing repeated coverage and saving energy. Our method can also shorten the traveling distance.

[1]This work is supported in part by the National Science Foundation IIS-0329061. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

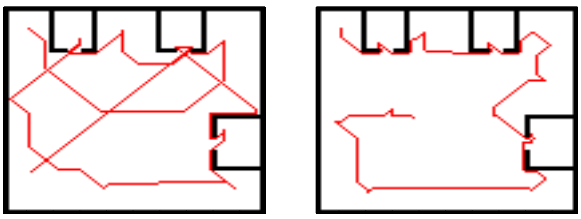
0-7803-9505-0/06/$20.00 ©2006 IEEE

II. Related Work

There are many studies on mobile robot exploration. Engelson [8] focuses on passive exploration where the robot motion is controlled by an operator. Chang et al. [5] compare energy and time efficiency of different dispatching algorithms for ant-like robot systems to explore an unknown area. For autonomous robot exploration, the key step is in selecting the next exploration target automatically. Yamauchi [15] proposes frontier-based exploration. The candidates of next targets are along the frontiers between the known area and the unknown area. Frontier-based methods have been used in some later studies [4] [13] [17]. Simmons et al. [13] propose an approach to coordinate multi-robot exploration. Robots submit their own estimations of the utilities and costs of different frontier cells, and a central control assigns targets to different robots to maximize the total utility. Zlot et al. [17] present a market control architecture for multi-robot exploration to minimize the cost and maximize the utility. Burgard et al. [4] present a method for coordinating multiple robots in exploration. Their method estimates the utilities of frontier cells in a coordinated way. The more robots move toward the same location, the lower the location's utility is, thus preventing multiple robots moving into the same place. This method requires communication among robots. These studies all adopt a utility-based strategy, selecting next target that can immediately maximize the utility and minimize the cost.

Several studies propose using markers in exploration. Batalin et al. [3] present a method of using markers for exploration and avoiding the localization problem. The robot drops markers to identify those places that have been explored. However, this method is limited by the number of available markers. Trevai at al. [14] distribute observation points into the environment using reaction-diffusion equations. The exploration task is reduced to traveling through all the observation points (markers). The disadvantage of this method is the requirement of distributing observation points before exploration.

Energy-efficient motion planning has been widely studied. Katoh et al. [10] present an energy-efficient motion planning method for space manipulator. This method controls the motion of the space manipulator to be elliptic. Mei at al. [12] build the power model of a robot at different speeds and consider stops and turns. Barili et al. [2] demonstrate the control of speed and the avoidance of stops in energy conservation for mobile robots. Jia et al. [9] propose a cost-efficient motion planning algorithm. The cost can be distance or time. However, their algorithm does not consider the robot's direction. Zelinsky et al. [16] propose a combination of quad-tree environmental representation and distance calculation for motion planning in an unknown environment.



(a) (b)

Fig. 1. (a) Utility-based target selection. (b) Orientation-based target selection.

None of the above studies focuses on energy-efficient exploration. Our study adopts an orientation-based method in selecting next targets and plans the path between the current location and the selected target energy-efficiently. Simulation results show that our energy-efficient exploration method can save up to 42% energy compared with one utility-based method.

III. Motivating Examples

A. Target Selection

Figures 1(a) and 1(b) show two exploration routes of the same robot exploring the same area. The routes are generated by our simulator to be described later. Inside this area, there are three rooms with small openings at the doors. The robot starts from the upper left corner and stops until the whole area is covered by robot's sensors. In Figure 1 (a), the robot selects the next target to maximize the utility, therefore skipping small openings at the doors or corners at the beginning. However, the robot has to visit the rooms or corners later to fully cover this area. There are many crossovers along the path, resulting in repeated coverage. In Figure 1(b), the robot selects the next target based on the relative directions of the frontier cells to the robot. When the robot moves, it always chooses the next target from the frontier cells on the left side first. If there is no frontier cell in the left side, it will choose frontier cells in the front. The priorities of the frontier cells depend on their relative directions to the robot, starting from robot's left side and following the clockwise sequence: left, front, right, and back. The essence of this strategy is to make sure that the robot's left side has been explored when the robot explores unknown areas in the front. Adopting this strategy, the robot in Figure 1(b) visits the rooms earlier than the robot in Figure 1(a). The strategy of Figure

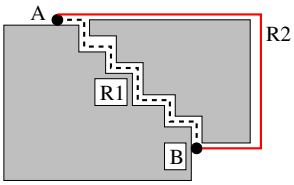


Fig. 2. Two routes R1 and R2 connecting location A with location B. R1 is shorter than R2, but consumes more energy.

1(b) is better because the path has no crossing point, and the path length is shorter. This example shows the importance of orientation-based target selection.

B. Energy-Efficient Motion Planning

Figure 2 shows the two routes from location 'A' to location 'B'. The gray area represents obstacles. Route R1 comprises of ten short line segments, while route R2 has three long line segments. R1 has a shorter distance with more stops and turns than R2. Stops and turns cause acceleration and deceleration that consume a significant amount of energy. Hence, R1 may be shorter but consume more energy. This shows the difference between the short-distance paths and energy-efficient paths.

IV. Energy-Efficient Exploration

A. Problem Definition

Exploration is to cover a 2-dimensional area by a robot's sensors. We use a grid cell map to represent this area. Each cell is a 1×1 unit of square. Each cell is either free or occupied by an obstacle. Obstacle cells are inaccessible to the robot and impenetrable to sensors. The robot can move from one cell to one of its eight neighbors, if both cells are free. If we set up a coordinate system, a cell can be represent by its two coordinates (i, j), where i and j are two nonnegative integers. We count a robot's movement from one cell to one of its neighbors as one step. In Figure 3 (a), there are eight free cells and one obstacle cell. From the cell in the center a robot can travel to seven out of the eight neighbors as illustrated by the arrows. At step 0, the robot starts from the initial location and senses the environment. The robot is equipped with sensors and the sensing range is a circle with a radius of d_s. This radius is also called the sensing distance. The robot moves and updates the explored map until all the accessible area has been explored. At each step, the robot's state can be represented by its location (i, j) and direction θ, and we denote robot's state at step k as $State(k) =< i(k), j(k), \theta(k) >$. The exploration

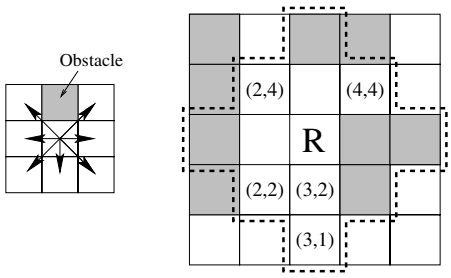


Fig. 3. (a) Free and obstacle cells. (b) R: robot's current location; area enclosed by dash line has been explored.

trajectory is a link of the robot's states at each step $State(0), ..., State(k), ...$

The energy-efficient exploration problem is to determine a trajectory to explore the whole accessible area with minimum energy. There are two sub-problems involved: target selection and motion planning. Target selection is to select a cell from frontier cells for the robot to explore next. Frontier cells are explored free cells along the border between the explored area and the unexplored area. Motion planning is to plan a viable path from the current cell to the target cell. Since both the current and the target cells are explored free cells, there must exist a viable path within the explored area. The robot continues target selection and motion planning until the end, and the exploration trajectory is thus generated.

B. Target Selection

Target selection determines which frontier cell to explore next. For example, in Figure 3 (b), the robot is at location $(3, 3)$, and all cells enclosed by the dash lines have been explored. There is a total of five frontier cells: $(2, 2)$, $(3, 1)$, $(3, 2)$, $(2, 4)$, and $(4, 4)$. Among the five frontier cells, three are connected: $(2, 2)$, $(3, 1)$, and $(3, 2)$. The utility-based method selects a frontier cell that can cover more potentially unexplored area. For example, in Figure 3 (b), the cell $(3, 1)$ is more distant from obstacles than the other four frontier cells; thus, the robot can cover more unexplored area after moving to this cell. A utility-based method selects $(3, 1)$ as the next target. However, as we have shown by the motivating example in section III-A, the utility-based method causes more repeated coverage.

Our method uses a orientation-based target selection strategy, as described in the following steps:

(1) It identifies all the frontier cells that are within the current sensing region. If no such frontier cell exists, go to step (4).

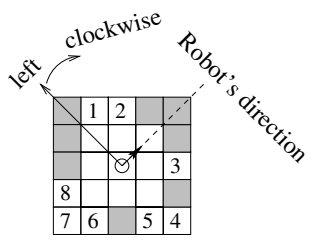


Fig. 4. Frontier cells and target selection. Our algorithm lists the 8 frontier cell in the number order: cell 1, cell 2, ..., cell 8, starting from robot's left direction and following the clockwise order.

(2) The algorithm lists all the frontier cells from step (1) in a clockwise order starting from the robot's left direction. Figure 4 illustrates this ordering. The figure shows 25 explored cells, and the area outside this region is unexplored. There are 8 frontier cells as labeled by the numbers. The cell at the robot's left direction is an obstacle cell. The 8 frontier cells are labeled from 1 to 8 in clockwise order; therefore, the list for this figure is cell 1, cell 2, ..., cell 8. The head of the list is cell 1.

(3) The algorithm picks a frontier cell in the list that satisfies the following two conditions: (a) From the list head to this frontier cell, any one cell and its next cell are neighbors. In other words, there is no jump along the list between the head to this cell. (b) The distance from the head to this cell is less than $0.7d_s$. The first condition promises that there is no obstacle between the head and the selected target cell along the border. The second condition assures that when the robot moves to the target, the robot can sense the obstacles in the left side and also some distance (at least $0.3d_s$) outside the head of the frontier cell list. This strategy is to make sure that the robot's left side has been explored when the robot proceeds to explore unknown areas in the front, essential to our orientation-based method.

If the algorithm does not follow the second condition, and directly picks the list head as the next target, the robot tends to move too close to obstacles. For example, in Figure 5 there are continuous obstacles (a wall). Picking the list head as the next target, the robot moves to a frontier that is a neighbor to the wall and then moves along the wall. Since the sensors can sense a distance of $d_s > 1$, this wastes the sensors' capability.

(4) If no frontier cell is within the current sensing region, the algorithm picks the closest frontier cell outside the current sensing range as the next target.

(5) If there is no frontier cell at all, then all the accessible area has been explored and the exploration is completed.

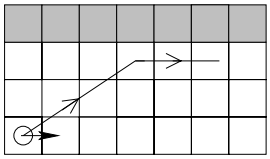


Fig. 5. Closely move along a wall.

C. Motion Planning

If the next target is within the robot's current sensing range as in step (1) of our target selection method, the current location and the target are close and a simple motion planning algorithm is sufficient. However, if there is no frontier cell inside the current sensing region, the robot may select a frontier cell far away form the current location as the next target, as in step (4) of our target selection method. In this situation, motion planning is important to find an optimal route.

Most existing studies plan a shortest path between the current location and the next target. Dijkstra's algorithm can find shortest paths for graphs. To find shortest paths using Dijkstra's algorithm, the grid cell map can be transformed into a graph in this way: free cells are vertices and edges exist between two vertices if they are neighbors. The edge has a weight of either 1 or $\sqrt{2}$ depending on their relative locations, representing the distance between two neighbor cells. The graph is undirected.

To use Dijkstra's algorithm to generate the energy-efficient paths, we transform the grid map into a graph in a different way. To incorporate the direction information, the vertices in the graph should represent the robot's states that include both locations and directions. Each free cell in the grid map is transformed into 8 vertices, representing the 8 possible robot states at this cell. If the cell is (i, j), the 8 vertices are 8 states: $< i, j, 0^\circ >$, $< i, j, 45^\circ >$, ..., $< i, j, 315^\circ >$. We assume the robot uses 45° as the unit for turns, since we only allow the robot to move from one cell to one of its eight neighbors. We can also label these 8 vertices by their directions: N in short of North, NE in short of Northeast, E, SE, S, SW, W, and NW. The cell has also 8 neighbor cells, and we can label them similarly. Each of these 8 neighbor cells are represented by 8 vertices, as 8 possible leaving states at that cell. Figure 6 shows two cells (i, j) and its Northeast neighbor $(i + 1, j + 1)$ with 8 vertices for each.

An edge connects two states. The edge is directed, because the robot may travel from one state to another state but not in the reverse direction. From any one state,

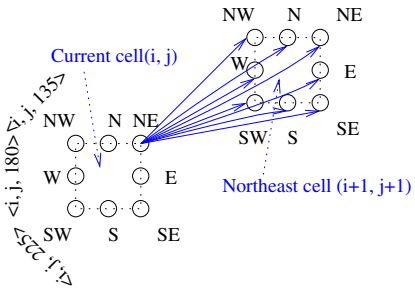


Fig. 6. Transform a grid cell map into a graph for energy-efficient motion planning. The circles are vertices, and the solid lines with arrows represent directed edges. The vertices represent the robot's states. The weight of one edge is the energy needed for the robot traveling from one state to another state.

the robot can only reach 8 other states. For example, from state $< i, j, 45^\circ >$, the robot can reach its Northeast neighbor $< i+1, j+1 >$, and this neighbor cell has 8 possible leaving states: $< i+1, j+1, 0^\circ >$, ..., $< i+1, j+1, 315^\circ >$. There are 8 edges that start from $< i, j, 45^\circ >$ and end at $< i+1, j+1, 0^\circ >$, ..., $< i+1, j+1, 315^\circ >$, respectively. In Figure 6, the solid lines with arrows show the 8 edges. The weight of one edge between two states is the energy needed for the robot to move from one state to the other state. We consider the energy for stops and turns if the two states have different directions. For example, the weight of the edge from 'NE' of (i, j) to 'NE' of $(i+1, j+1)$ is only the energy of traveling a distance of $\sqrt{2}$, because the robot does not stop or turn. However, the weight of the edge from 'NE' of (i, j) to 'E' of $(i+1, j+1)$ includes the energy for traveling distance $\sqrt{2}$ and the energy for a stop and a turn of 45°.

The above two paragraphs explain how to transform a grid map into a graph. In such a graph, a path represents a trajectory of the robot and the sum of the weights of the edges along a path represents the energy consumption of that path.

V. Simulations and Results

A. Simulation Setup

We use the following parameters in our simulations. The cell is a square with each side of one unit length. The robot consumes one unit of energy for traveling one unit of distance. One stop takes an extra energy of 0.5 unit. A turn of 45° takes 0.4 unit of energy. Turns of 90°, 135°, 180° take 0.6, 0.8 and 1 unit of energy, respectively. These numbers are approximately derived from our energy measurements for a Pioneer 3-DX robot [1]. The robot's sensing range is a circle with a radius $d = 10$ units of distance. The unknown areas to be explored are rectangles of different sizes. Two types of obstacles are used for simulations: random obstacles and structured obstacles. Figures 7, 8, 9, and 10 show random and structured maps in our simulations. In these figures, white cells represent free cells and black cells represent obstacle cells. These areas are bounded by obstacles.

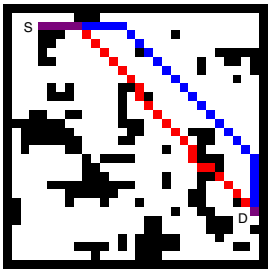


Fig. 7. Shortest and energy-efficient paths.

B. Energy-Efficient Motion Planning

We compare the paths generated by our energy-efficient motion planning method with the shortest paths. We use random maps for simulations. Figure 7 shows two paths from the source cell "S" to the destination cell "D". The lower one is the shortest path of length 33.87, consuming energy of 41.07 units. The upper one is the energy-efficient path with a length of 36.21, 6.9% longer than the shortest path. However, the energy-efficient path only consumes energy of 38.01 units, which is 7.5% lower than the energy consumption of the shortest path.

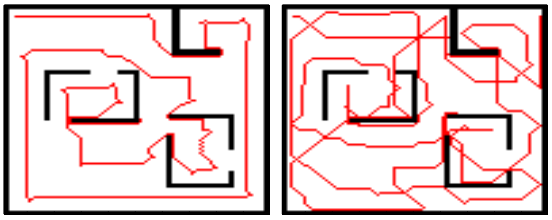
In a random map with 20% obstacle cells, we compare the distances and the energy consumption of the shortest and the corresponding energy-efficient paths between more than 3800 different pairs of source and destination cells. The results show that on average the energy-efficient paths save 8.4% energy while they are 0.7% longer compared with the shortest paths. This is because the energy-efficient paths have fewer stops and turns that may consume significant amounts of energy.

C. Target Selection and Robot Exploration

We run simulations and compare our method with a utility-based method [4], called "widest frontier". This method first clusters the frontier cells within the current sensing region into groups. The frontier cells inside one group are close to each other, and they are connected as neighbors. We call each group as one frontier. This method chooses a widest frontier, the group with the

maximum number of frontier cells, and picks one in the middle as the target cell. If no frontier cell is found in the current region, the algorithm picks one closest frontier cell outside as the next target. This method is based on utility of frontier cells because moving to the middle cell of the widest frontier is likely to cover more new area than any other frontier cell.

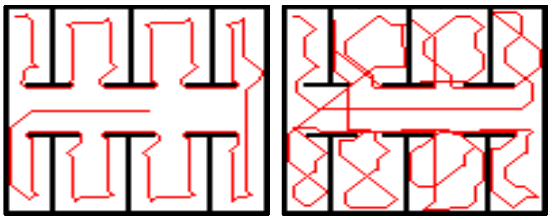
Figures 8 (a) and (b) show two different exploration routes of the same map generated by our target selecting method and the widest frontier method, respectively. The route from our method has a length of 710.8 with total energy consumption of 785.6. The route from the widest frontier has a length of 1222.6 with total energy consumption of 1374.3. The route from our method is 41.8% shorter in distance and consumes 42.8% less energy.



(a) (b)

Fig. 8. (a) Our target selection method. (b) Choose the widest frontier.

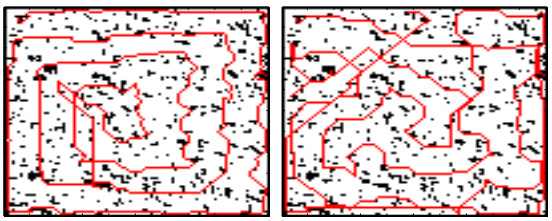
Figures 9 (a) and (b) show the exploration routes in another map with structured obstacles. The route from our method has a length of 798.3 with total energy consumption of 889.6. The route from the widest frontier has a length of 1308.3 with total energy consumption of 1445.7. The route from our method is 38.9% shorter in distance and consumes 38.5% less energy.



(a) (b)

Fig. 9. (a) Our target selection method. (b) Choose the widest frontier.

The above two maps are areas with structured obstacles. Figures 10 (a) and (b) show the exploration routes in an area with random obstacles. The route from our method has a length of 1075.5 with total energy consumption of 1261.3. The route from the widest frontier has a length of 1207.1 with total energy consumption of 1390.1. The route from our method is 10.1% shorter in distance and consumes 9.3% less energy.



(a) (b)

Fig. 10. (a) Our target selection method. (b) Choose the widest frontier.

From the above simulation results, we can see that our orientation-based target selection method can reduce repeated coverage, have shorter exploration distances, and consume less energy. We further investigate the coverage ratio. The coverage ratio is the number of explored free cells over the total number of accessible free cells. It is 0 when the robot starts and it is 1 at the end. If we study the intermediate coverage ratio, we expect that the widest frontier method is better at the beginning. This is because widest frontier method can cover more area at the beginning. However, the widest frontier method becomes worse later to cover many small unexplored places and crosses over many explored areas. Figure 11 shows the coverage ratios of the two routes corresponding to Figure 8. The widest frontier method leads before 450 steps and lags behind after that. Figure 12 shows the coverage ratios of the two routes corresponding to Figure 9. The widest frontier method leads before 230 steps and then lags behind.

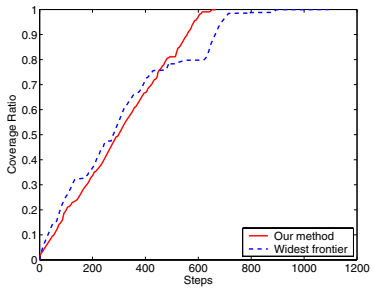


Fig. 11. Coverage ratio at different steps for Figure 8

Figure 13 shows the coverage ratios of the two routes corresponding to Figure 10. In this area with random obstacles, the widest frontier method leads most of the time, only to worsen after 830 steps. This can be explained as following. In a map with random obstacles, the obstacles are scattered and the robot can easily find wide frontiers to explore. However, in a structured area, the free space is divided into subspaces by large obstacles. After the robot finishes exploring large frontiers

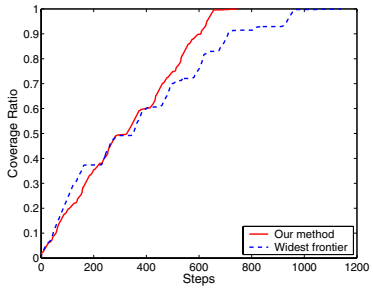


Fig. 12. Coverage ratio at different steps for Figure 9

in one subspace, it needs to travel to another subspace to explore large frontiers. Later, the robot has to revisit these subspaces to explore small frontiers. Therefore, our method are superior to the utility-based method in areas with structured obstacles.

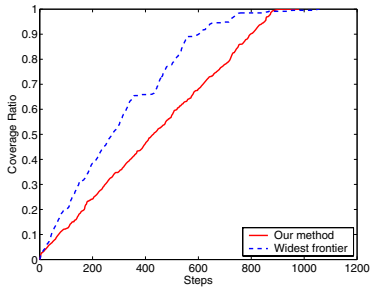


Fig. 13. Coverage ratio at different steps for Figure 10

VI. Conclusion

In this study, we have presented an approach for target selection and energy-efficient motion planning for robot exploration. We have compared our method with "widest frontier", one of the utility-based methods that represent the state of the art in robot exploration. Our method is orientation-based and avoids much repeated coverage that is common in utility-based methods. Simulation results show that our method save up to 42% energy and 41% traveling distance in areas with structured obstacles. For future work, we plan to compare our method with more utility-based methods in different types of environments, such as randomly generated structured maps, and extend our work to multi-robot exploration.

References

[1] D. Auguelov, D. Koller, E. Parker, and S. Thrun. Detecting and Modeling Doors with Mobile Robots. In *International Conference on Robotics and Automation*, pages 3777–3784, 2004.

[2] A. Barili, M. Ceresa, and C. Parisi. Energy-Saving Motion Control for An Autonomous Mobile Robot. In *International Symposium on Industrial Electronics*, pages 674–676, 1995.

[3] M. A. Batalin and G. S. Sukhatme. Efficient Exploration without Localization. In *International Conference on Robotics and Automation*, pages 2714–2719, 2003.

[4] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated Multi-Robot Exploration. *IEEE Transaction on Robotics*, 21(3):376–386, 6 2005.

[5] H. J. Chang, C. S. G. Lee, Y.-H. Lu, and Y. C. Hu. Energy-Time-Efficient Adaptive Dispatching Algorithms for Ant-Like Robot Systems. In *International Conference on Robotics and Automation*, pages 3294–3299, 2004.

[6] A. Davids. Urban Search and Rescue Robots: From Tragedy To Technology. *IEEE Intelligent Systems*, 17(2):81–83, March 2002.

[7] A. Drenner, I. Burt, T. Dahlin, B. Kratochvil, C. McMillen, B. Nelson, N. Papanikolopoulos, P. E. Rybski, K. Stubbs, D. Waletzko, and K. B. Yesin. Mobility Enhancements to the Scout Robot Platform. In *International Conference on Robotics and Automation*, pages 1069–1074, 2002.

[8] S. Engelson. *Passive Map Learning and Visual Place Recognition*. PhD thesis, Department of Computer Science, Yale University, 1994.

[9] M. Jia, G. Zhou, and Z. Chen. An Efficient Strategy Integrating Grid and Topological Information For Robot Exploration. In *IEEE Conference on Robotics, Automation and Mechatronics*, pages 667–672, 2004.

[10] R. Katoh, O. Ichiyama, T. Yamamoto, and F. Ohkawa. A Real-time Path Planning of Space Manipulator Saving Consumed Energy. In *International Conference on Industrial Electronics, Control and Instrumentation*, pages 1064–1067, 1994.

[11] C. Luo and S. X. Yang. A Real-Time Cooperative Sweeping Strategy for Multiple Cleaning Robots. In *International Symposium on Intelligent Control*, pages 660–665, 2002.

[12] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee. Energy-Efficient Motion Planning for Mobile Robots. In *International Conference on Robotics and Automation*, pages 4344–4349, 2004.

[13] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for Multi-Robot Exploration and Mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2000.

[14] C. Trevai, Y. Fukazawa, J. Ota, H. Yuasa, T. Arai, and H. Asama. Cooperative Exploration of Mobile Robots Using Reaction-Diffusion Equation on a Graph. In *ICRA*, pages 2269–2274, 2003.

[15] B. Yamauchi. A Frontier-Based Approach for Autonomous Exploration. In *ICRA*, pages 146–151, 7 1997.

[16] A. Zelinsky. A Mobile Robot Exploration Algorithm. *IEEE Transaction on Robotics and Automation*, 8(6):707–717, 1992.

[17] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-Robot Exploration Controlled by a Market Economy. In *ICRA*, pages 3016–3023, 2002.