ISSN: 2321-2152

IJMECE

International Journal of modern electronics and communication engineering

E-Mail
editor.ijmece@gmail.com
editor@ijmece.com

www.ijmece.com

ISSN 2321-2152
www.ijmece.com
Vol 12, Issue 4, 2024
https://zenodo.org/records/14506144

Strategy for a Robotic Team to Explore and Survey a Region

Marneni Swapna[1], Assistant Professor[1], Department of ECE, Siddhartha Institute of Technology & Sciences, Telangana, India

Soumya Mudumba[2], Assistant Professor[2], Department of CSE, Siddhartha Institute of Technology & Sciences, Telangana, India.

Abstract

The difficulty of many robots exploring and mapping a new area is discussed in the abstract of this article. The mapping method employs hill climbing to find maps that are optimally compatible with sensor data and odometry. It is an online approach to probability maximisation. The robots are explicitly coordinated by the exploring algorithm. It aims to optimise total value by reducing the likelihood of knowledge acquisition overlap across the different robots. The majority of calculations are spread for both the exploration and mapping methods. The methods have undergone considerable testing in simulations and real-world trials. The outcomes illustrate the resilience and performance enhancements that occur from our multirobot approach to exploration.

Introduction

Creating maps of the environment is a fundamental challenge in mobile robotics. In general, to do so efficiently requires good exploration strategies. In particular, the robots need to know what areas are worthwhile to explore and how to distribute themselves effectively in order to thoroughly map previously unknown areas. Most previous work in mapping dealt only with single robots. There are, however, advantages in mapping with multiple robots. The most obvious is that multiple robots can often do the task in less time. This may not always hold, however, due to interference between robots [6, 8]. Thus, it is important for the exploration strategies to keep the robots relatively well separated. Another advantage is that multiple robots may produce more accurate maps, due to merging of overlapping information. This can help compensate for sensor uncertainty and localization error, especially where the robots have different sensor and/or localization capabilities [7]. This paper presents techniques for coordinating multiple, heterogeneous robots in their task of exploring and mapping large, indoor environments. We consider two coordination problems — creating a single global map from the sensor information of the individual robots, and deciding where each robot should go in order to create the map most effectively. While solving the latter problem optimally is intractable, we present a greedy approach that performs quite well, in practice. Our basic approach to both coordination problems is similar: Distribute most of the computation amongst the individual robots and asynchronously integrate their results by performing some global computations over the data. For instance, each robot processes its own laser data to create a consistent local map. A central mapper module then integrates the local maps to create a consistent global map. The local mappers reduce uncertainty in the data, principally by matching laser scans to decrease localization error. The central mapper further improves the map (minimizing localization error) by iteratively combining data from the robots. This works under the assumption that the robots know their pose relative to one another and have access to high-bandwidth communication. Similarly, our approach to coordinating exploration combines distributed computation with global decision making.

The individual robots construct “bids,” which describe their estimates of the expected information gain and costs of traveling to various locations. A central executive receives the bids and assigns tasks in an attempt to maximize overall utility, while trying to minimize overlap in coverage by the robots. In both cases, the majority of the computation is done in a distributed fashion, by the individual robots, and the centralized modules combine and coordinate information in an efficient way. After presenting related work, Sections 3 and 4 describe our approaches to multi-robot map creation and exploration, respectively. Section 5 presents a case study of three robots combining to map a large indoor area. We also analyse quantitative results from simulations showing the effects of our exploration strategies on task performance. Finally, we discuss future directions that are important to the problems of multi-robot exploration and mapping.

Related Work

While there has been work in mapping and exploration for single robot systems [3, 4, 9, 17, 18], there have been relatively few approaches for mapping and exploration with multi-robot systems. Several researchers have studied the problem of using multiple robots to reduce localization Copyright 2000, American Association for

ISSN 2321-2152
www.ijmece.com
Vol 12, Issue 4, 2024
https://zenodo.org/records/14506144

Artificial Intelligence error during exploration [10, 13]. For instance, in Realities. al. [13] the environment is divided into strips. Each strip is explored by a single robot, while the others remain stationary to observe the moving robot and estimate its position. While this has the advantage of improving the overall accuracy of the map, it does nothing to speed the exploration process. On the contrary, the robots are forced to remain near each other in order to stay visible. Balch and Arkin [1] investigated how communication in multi-robot systems affects different tasks, including the graze task where the objective is to completely cover an unknown environment. The robots essentially perform a randomized search of the environment. Their performance results are qualitatively similar to what we observe, but we have not done any direct comparisons of the two methods. More sophisticated techniques for multi-robot exploration are presented in [15, 21, 22]. Singh and Fujimura [15] present a decentralized on-line approach for heterogeneous robots. When a robot discovers an opening to an unexplored area that it cannot reach because of its size, it selects another robot which can carry out the exploration task. The candidate robot is chosen by trading off the number of areas to be explored, the size of the robot, and the straight-line distance between the robot and the target region. Yamauchi developed a technique in which the robots build a common map (an occupancy grid) in a distributed fashion [21, 22]. The work introduces the notion of a frontier, which is a location near an unexplored part of the environment.

The approach groups adjacent cells into frontier regions. Each robot then heads for the centroid of the closest frontier region, but they do so independently — while they share maps, there is no explicit coordination. Thus, the robots may end up covering the same area and may even physically interfere with one another. Our approach, in contrast, tends to keep the robots well separated, which can significantly decrease the time needed to accomplish the mapping task. The work reported here extends our earlier efforts [2] in several important ways. First, the approach described here distributes the computation, to a large extent. This enables the robots' "bids" to be calculated in parallel, which facilitates scaling to larger numbers of robots and enables the robots to construct bids based on their own capabilities (sensor range, travel costs, etc.). Second, the current method uses a more sophisticated notion of expected information gain that takes current map knowledge and the robots' individual capabilities into account. This allows for more subtle types of coordination, for example, allowing the robots to remain near one another if the map shows that they are separated by a solid wall.

Coordinated Mapping

At the core of our approach is a distributed algorithm for concurrent mapping and localization in real-time [19]. The approach makes two major assumptions: First, it assumes the world is reasonably static, and so it cannot handle

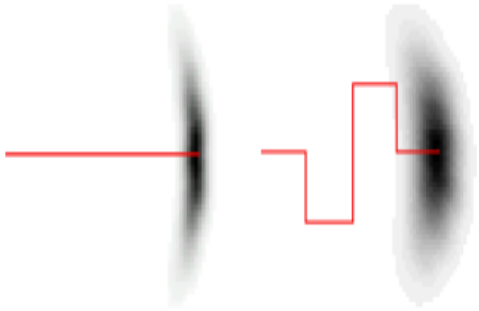

Figure 1: Probabilistic Motion Model Robot

starts at left of each diagram and follows path indicated by solid line. Probability distribution is shown in grey for the robot's posterior location. The darker a location, the more likely it is environments that are densely populated or change in major ways (e.g., walls disappearing). Second, it assumes that the robots begin in view of one another, and are told their approximate relative location (within about 1 meter distance and 20 degrees orientation). The first requirement is assumed throughout the literature on concurrent localization and mapping [3, 12, 16, 18]. Fortunately, the second assumption holds for many practical applications, since the problem of team-based mapping in the absence of initial pose information is extremely hard. Our approach decomposes the mapping problem in a modular, hierarchical fashion: Each robot maintains its own local map, correcting for odometry error as it goes. A central module receives the local maps and combines them into a single, global map. The modules work in real-time and, in fact, adapt their computational requirements to the available resources.

The beauty of the approach is that basically the same software runs at both the local and global levels. To start, each robot receives a sequence of its own odometry and sensor measurements (laser range scans, in our case). From that, it incrementally constructs three things: a maximum

ISSN 2321-2152
www.ijmece.com
Vol 12, Issue 4, 2024
https://zenodo.org/records/14506144

likelihood estimates for its own position, a maximum likelihood estimates for the map (location of surrounding objects), and a posterior density characterizing its "true" location, which acknowledges the fact that certain errors cannot be identified when building a map [20]. To illustrate the algorithm, assume that a robot has already developed a partial map. It now wants to augment the map through new sensor and odometry readings. To determine the robot's most likely position, our algorithm maximizes a mixture likelihood function that model (1) the noise in motion (odometry), and (2) the noise in perception. Figure 1 illustrates the motion model. It depicts P (s | s', a), the probability of being at pose s, if the robot was previously at s' and executed action a (moving and/or turning). This distribution is obtained by the (obvious) kinematic equations, assuming that robot motion is noisy along its translational and rotational components. Figure 2 depicts the perceptual model (the likelihood function for sensor readings). The basic idea here is that it is unlikely to receive sensor readings where previous scans saw free-space. The dark region in Figure 2 corresponds to

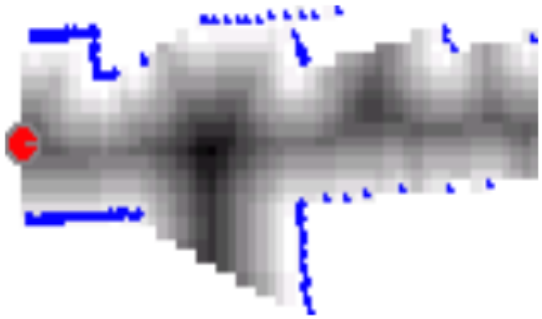

Figure 2: Likelihood Function Generated

Robot is on the left (circle). The scan is depicted by 180 dots in front of the robot. The darker a region, the smaller the likelihood for sensing an object there. Occluded regions are white.

the free-space of the scan shown there; the likelihood of detecting something in that region is (inversely) proportional to the grey-level. Thus, scans that nicely align possess much higher likelihood than ones that do not. A key characteristic of this likelihood function it that it is differentiable. Moreover, search in the relative pose space of the robot can be performed very efficiently using Newton's method (e.g., 1,000 iterations per second). Our approach starts with the odometry measurement reported by the robot as an initial estimate, and uses gradient descent to find the nearest maximum in likelihood space. Since maps are built incrementally and short-term errors are not large, this process converges quickly and, with high reliability, finds the right alignment. The collection of all scans, along with their corrected scan coordinates, forms the map. The scan map is then efficiently converted to an occupancy grid map [5], which is required by our motion planner and exploration module. We now address the problem of building a map using multiple mobile robots. Each robot builds its own local map, using the algorithm described above. Since the robots do not communicate directly, their local coordinate systems are not aligned with each other. Also, due to residual errors in the local maps, the maps typically would not match well even if the coordinate systems were perfectly aligned. To build a single map, the central mapper module integrates information from the individual robots in real time. Specifically, each robot communicates a subset of its scans (e.g., every 10th) to the central mapper, using the corrected scan coordinates. Thus, the maps of the robots are not used directly. Instead, they are used indirectly to produce sequences of scans whose (relative) position errors are already very small to begin with. The central mapper then applies the same gradient descent algorithm described above to minimize the error between the scans of the different robots. Since we assume that the initial positions of the robots are approximately known, our local search approach accurately localizes the robots relative to each other. As additional scans arrive, they are similarly mapped into the global coordinate system, eliminating small deviations. The

Figure 3: Occupancy Map Used for Exploration "Obstacle" cells are black, "clear" cells are white, "unknown" cells are grey. Frontier cells are marked by small circles.

resulting map integrates every robot'sscan into a single, consistent map with relatively little

ISSN 2321-2152
www.ijmece.com
Vol 12, Issue 4, 2024
https://zenodo.org/records/14506144

computation. We have tested our procedure for up to 5 robots, and have no doubt that the same architecture easily scales to 10, or more, robots.

Coordinated Exploration

The objective in coordinating the exploration of multiple robots is to maximize expected information gain (map knowledge) over time. While the optimal solution is computationally intractable, we have developed a relatively low-cost technique that provides good results, in practice. To start, each robot constructs a "bid" consisting of the estimated utilities for it to travel to various locations. The bids are sent to a central executive, which assigns tasks to each robot based on all the bids received, taking into account potential overlaps in coverage. Thus, while a robot may prefer to visit one location, the executive might assign it a different location if another robot is expected to gain much the same information. Robots submit new bids when their maps are updated, which can cause them to be retacked. Exploration ends when there is no useful information to be gained.

Constructing Bids

A robot constructs a new bid each time it receives a map update from the central mapper. It categorizes map cells into three different types (Figure 3) — "obstacle" (probability of occupancy above a given threshold po), "clear" (probability below a threshold pc) and "unknown" (either never been sensed, or probability is between po and pc). A bid is a list of the estimated costs and information gains for visiting various frontier cells. We define a frontier cell as any "clear" cell adjacent to at least one "unknown" cell (Figure 3). For efficiency, we further stipulate that each frontier cell must be at least some minimum distance from all other frontier cells. For instance, even though our grid has 15 cm resolution, we require frontier cells to be at least 30 cm (approximate radius of the robots) from each other. To estimate the cost of visiting a frontier cell, we compute the optimal path (shortest distance, assuming deterministic motion) from the robot's current position. All costs are

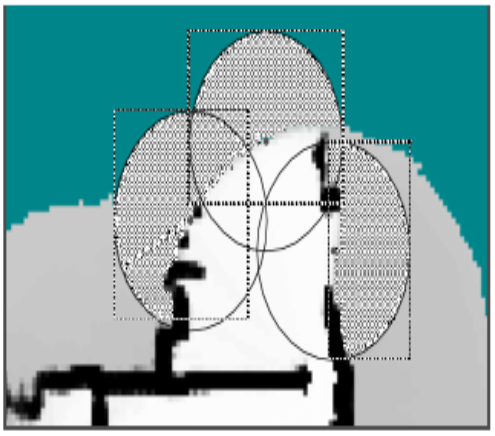

Figure 4: Expected Information Gain Information gain regions for several representative frontier cells. Circles indicate sensor range. Cross-hatched areas are information gain regions. Dotted lines are the rectangular approximations.

computed simultaneously, using a simple flood-fill algorithm [11] that employs an efficient implementation of a priority queue to propagate minimum path costs through the map. To further decrease computation, we consider only "clear" cells, stopping propagation whenever an "obstacle" or "unknown" cell is reached. Estimating information gain is more difficult. In fact, the actual information gain is impossible to predict, since it very much depends on the structure of the environment. Our previous work [2] assumed information gain to be constant for each frontier cell, which tended to make the robots spend too much time exploring nearby areas that were nearly known already. Here we use the current map to provide a more informed estimate. Specifically, we assume that the robot has some nominal sensor range and count the number of "unknown" cells that fall within that radius of the frontier cell, subject to the restriction that the resulting information gain region forms a connected set (Figure 4). For efficiency, we again use a flood-fill algorithm, this time ending propagation when either a "clear" or "obstacle" cell is encountered, or when the distance to the frontier cell is greater than the sensor range. In addition to counting the number of cells, we record the minimum and maximum extent, forming a rectangle that approximates the information gain region (Figure 4).

This enclosing rectangle is used by the executive to estimate potential overlaps in coverage. While there are definite improvements that can be made in estimating information gain (a simple one would be to bias the count by the occupancy probability of

ISSN 2321-2152
www.ijmece.com
Vol 12, Issue 4, 2024
https://zenodo.org/records/14506144

the cells, giving less weight to cells that are already partially known), we have found our metric works quite well in practice. In particular, while the metric usually acts to keep the robots well separated, it still allows them to be spatially near one another if there are known obstacles separating them. Thus, we have seen cases where two robots are tasked to explore adjacent rooms — one goes into the first room and perceives the walls, which get added as obstacles in the map. This separates the room from the adjacent room, informationwise, which allows the executive to send the other robot into the second room. This would not be possible with methods that merely try to maintain a given distance between robots.

Assigning Tasks

Each robot asynchronously constructs its bids and sends them to a central executive. The executive tries to maximize the total expected utility of the robots by assigning them tasks, based on their bids. A simple greedy algorithm is used to keep the computation real time. The executive first finds the bid location with the highest net utility (information gain minus cost) and assigns that task to the robot that made the bid. 1 It then discounts the bids of the remaining robots based on the current assigned tasks (see below) and chooses the highest remaining net utility. This continues until either all robots have been assigned tasks or no task remains whose (discounted) expected information gain is above a minimum threshold. Key to this algorithm is the discounting. Without it, the robots would act in an uncoordinated manner, being assigned tasks that they, independently, estimate as best. Our previous work discounted the utility of a location as a function of its distance to the other assigned tasks [2]. Here, we explicitly use the estimated information gain for discounting. Specifically, we estimate the percentage of overlap between the information gain regions by how much the approximating rectangles overlap (Figure 4) and decrease the expected information gain by that percentage:

$$d_j = Area\left(IGR_j \cap \left(\bigcup_{i \in R} IGR_i\right)\right) / Area(IGR_j)$$

$$u_j = (1 - d_j) \times i_j - c_j$$

Here, IGRj is the rectangular approximation of the information gain region for some frontier cell j, the IGRi are the rectangular information gain regions for the assigned robots R, and i j and cj are the expected information gain and path cost of going to cell j. This method is both efficient to compute and a fairly accurate approximation. In one set of experiments, it was within 15% of the true overlap (obtained by counting the actual number of overlapping cells), while being hundreds of times more efficient. The executive is implemented using the Task Description Language (TDL), an extension of C++ that includes syntactical support for hierarchical task decomposition, task sequencing, execution monitoring, and exception handling [14]. When the executive receives a bid, it waits a short while in case other bids arrive. It then assigns tasks to all robots that are either currently unassigned or have submitted new bids (leaving the currently active ones to continue).

Figure 5: Robin, Marian and LittleJohn

Besides assigning tasks, the executive monitors task execution, interfaces with a remote GUI, and interleaves exploration with other tasks. In particular, at any time the user (through the GUI) can request that one of the robots visit a particular location (e.g., to take a closer look). The executive terminates that robot's current task, reassigns other robots to cover for its loss, assigns and monitors the new task, and then integrates the robot back into the exploration pool when it is finished. One important addition to the task assignment algorithm is the use of hysteresis. If a frontier cell for a robot falls within the information gain region of the robot's currently assigned task, then its expected information gain is divided by the hysteresis ratio (a constant between 0 and 1, usually 0.85). Lower values for the hysteresis ratio will make the executive less disposed to switching tasks. The basic problem is that, because the robot is continuously sensing the environment, the information gain metric can change drastically after only small motions. For instance, by the time a robot maneuvers to position itself in front of a doorway, it typically has seen a large portion of the

ISSN 2321-2152
www.ijmece.com
Vol 12, Issue 4, 2024
https://zenodo.org/records/14506144

room. Without hysteresis, entering and completely exploring the room would not have as much utility as going somewhere else. While not the ultimate solution, hysteresis handles the problem fairly well.

Experiments

The multi-robot exploration and mapping system has been tested extensively using a team of three heterogeneous robots — two Pioneer AT robots from RWI and an Urbie robot from IS Robotics (Figure 5). All three robots are equipped with Sick laser scanners that have a 180-degree field of view. The ATs have a 300 MHz on-board laptop running Linux, and all three robots communicate, via Breezecom radio links, with off-board Linux workstations that run the rest of the system, including the mapping, planning, and executive modules. The most extensive testing was in October, 1999 in an empty hospital building at Fort Sam Houston, San Antonio TX, as part of DARPA's Tactical Mobile Robot (TMR) project. During a five day period, we made repeated runs with the robots, mapping large areas of one floor of the building.

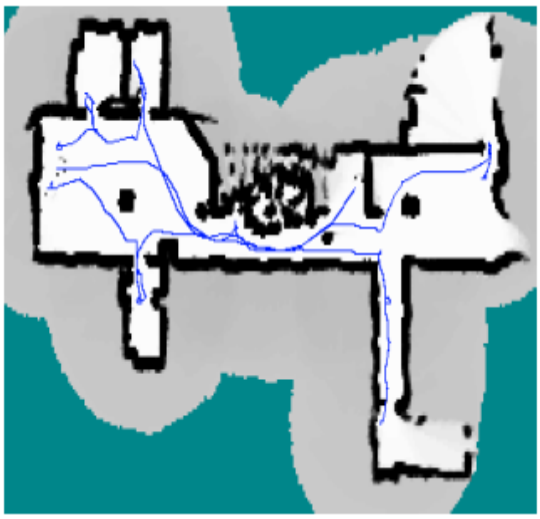

Figure 6: Map Created by Three Robots (62 x 43 m) Robots start at left. The three solid lines indicate the robots' paths through the environment.

Figure 6 shows one typical run that produced a 62 by 43 meter map in about eight minutes. During these runs, we tended to see similar qualitative behavior — one robot would head down the initial corridor, while the other two would explore rooms on opposite sides of the corridor. When the two finished the initial set of rooms, they would move down the corridor to explore openings that the third robot had discovered, but passed by. We also performed tests where we would teleoperate one of the robots while having the other two autonomously explore the areas not visited by the first. Some interesting behaviors were observed that are attributable to the coordination algorithm. For one, if three robots start in the middle of a narrow corridor, two tend to head down the corridor in opposite directions, while the third just waits until one of the others spots a doorway. This is because, initially, there are just two distinct frontiers, and assigning one robot to each leaves the third with no expected information gain. Another behavior, noted earlier, is that the robots sometimes explore adjacent offices that, while spatially close, are disconnected in terms of information gain. Finally, in one instance, we noticed a robot having trouble getting near an office it was tasked to explore. A second robot was tasked to explore further down the corridor. However, at some point the executive swapped tasks, since the second robot had fortuitously gotten closer to the office than the first. Such flexibility in dynamically coordinating tasks gives our system the ability to efficiently explore in a wide variety of situations. To augment the robot tests, we ran experiments in simulation to compare the effects of different numbers of robots in different types of environments. The simulator realistically models the environment and a robot's interaction with it, so that the programs used on real robots can be used with the simulator without modification.

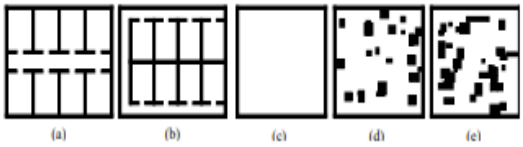

Figure 7: Simulation Environments

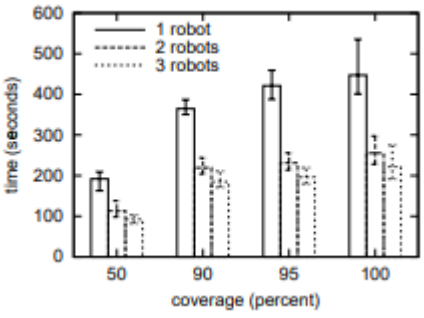

Figure 8: Results from Single-Corridor Environment

Previously, we demonstrated the performance increase that obtains using coordinated versus uncoordinated robots [2]. In the current experiments, we varied the number of robots from

ISSN 2321-2152
www.ijmece.com
Vol 12, Issue 4, 2024
https://zenodo.org/records/14506144

one to three, and used five different environments. In the two office-like environments (Figure 7, A and B, 25m x 20m) and the obstacle-free environment (C, 20m x 20m), we ran ten simulations for each number of robots. For the other two environments (D and E, 20m x 20m), we used ten different randomly generated maps, and ran one simulation with each. While our primary performance metric is the time needed to completely explore the environment, it is also of interest to see how the coverage evolves over time. It might be the case that most of an environment is mapped quickly, while it takes a long time to cover a few last spots at the end. For this reason, we report the time it takes to cover 50, 90, 95 and 100 percent of the environment. Figure 8 presents the results for the single-corridor office environment (A). It shows that two robots perform significantly better than one, while there is not much gain in having three robots instead of two. The results are similar for two parallel corridors (B). There, two robots can go in separate directions at the beginning, each exploring one part without any overlap. While a third robot can assist initially in the exploration of one of the corridors, once done it must travel a long way in order to help explore the other corridor. In many cases, the other robots do not arrive in time to help out. In contrast, in the random environments, there is a smaller gain when going from one to two robots and a larger gain when going from two to three robots (Figure 9), compared with the results from the office-like environments. The apparent reason is that the obstacles in the environment help in spreading the robots out.

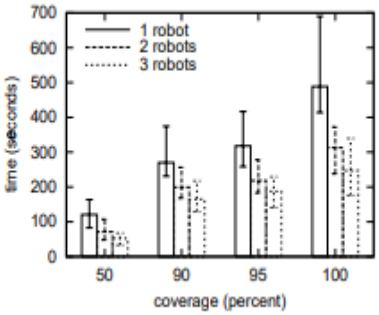

Figure 9: Results from 15% Random Obstacles

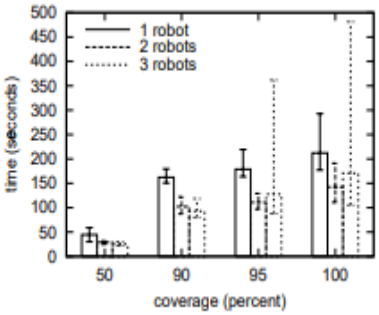

Figure 10: Results from Obstacle-Free Environment

Surprisingly, in the obstacle-free environment, three robots actually take longer to complete the task, on average, than two (Figure 10). This seems to be because they end up interfering with one another [6, 8]. In contrast, in this environment multiple robots have demonstrable positive effects on map accuracy. With few features, the robots get little help in localizing. This has the greatest impact when there is only one robot exploring — in fact, in 30% of the trials the robot failed to complete the mapping task successfully (i.e., the resultant map was qualitatively wrong). With multiple robots, however, the added sensor information helps significantly: For two robots, only one failure was observed (10%) and with three robots, no failures were observed.

Conclusions

While we have extensive test results, we still need to quantify the effects of various design decisions, including the effects of hysteresis, the way information overlap is estimated, and the definition of expected information gain itself. We also intend to quantify the performance of the greedy method of task assignment, comparing it to more sophisticated algorithms such as A* or stochastic search. In both simulation and actual tests, robots are sometimes idle because their discounted utilities are below the minimum threshold. Instead of just staying where they are, they could position

https://zenodo.org/records/14506144

themselves strategically so as to minimize the expected distance they would have to travel once they are assigned a task. While for a single robot, a good idle location is one that minimizes the average path cost to all the frontier cells, the problem is much harder if there are several idle robots. Fundamentally, the approach described in this paper is limited in two respects. First, with respect to mapping, we currently assume that the robots begin in view of one another and are told their initial (approximate) relative location. More sophisticated techniques are needed for mapping and localization when the robots need to merge maps where the coordinate transform is initially unknown and the robots need to find out where they are relative to one another. Second, with respect to exploration, we currently assume it is sufficient to consider the utility of exploring a single point. The approach ignores both the fact that information is gained en route and that moving to a given area may facilitate, or possibly hinder, subsequent exploration. We are investigating more sophisticated algorithms that estimate information gain along paths, which we believe will improve overall performance significantly

References

[1] T. Balch and R.C. Arkin. "Communication in Reactive Multiagent Robotic Systems." Autonomous Robots 1, pp. 1-25, 1994.

[2] W. Burgard, D. Fox, M. Moors, R. Simmons and S. Thrun. "Collaborative Multi-Robot Exploration." In Proc. Intl. Conf. on Robotics and Automation, San Francisco CA, May 2000.

[3] H. Choset. Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph. Ph.D. Thesis, California Institute of Technology, 1996.

[4] G. Dudek, M. Jenkin, E. Milios and D. Wilkes. "Robotic exploration as graph construction." IEEE Transactions on Robotics and Automation, 7:6, pp. 859-865, 1991.

[5] A. Elfes. Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation. Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

[6] M. Fontan and M. Mataric. "Territorial Multi-Robot Task Division." IEEE Transactions on Robotics and Automation, 14:5, 1998.

[7] D. Fox, W. Burgard, H. Kruppa and S. Thrun. "Collaborative Multi-Robot Localization." In Proc. 23rd German Conf. on Artificial Intelligence. Springer-Verlag, 1999.

[8] D. Goldberg and M.J. Mataric. "Interference as a Tool for Designing and Evaluating Multi-Robot Controllers." In Proc. AAAI-97, pp. 637-642, Providence, RI, July, 1997.

[9] B. Kuipers and Y.-T.Byun. "A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations." Journal of Robotics and Autonomous Systems, 8, pp. 47-63, 1991.

[10]R.Kurazume and N. Shigemi. "Cooperative Positioning with Multiple Robots." In Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), 1994.

[11]J.C. Latombe. Robot Motion Planning. Kluwer Academic Publishers, 1991.

[12]J. Leonard, H. Durrant-Whyte and I. Cox. "Dynamic Map Building for an Autonomous Mobile Robot." International Journal of Robotics Research, 11:4, pp. 89-96, 1992.

[13]I. Rekleitis, G. Dudek, E. Milios. "Accurate Mapping of an Unknown World and Online Landmark Positioning." In Proc. of Vision Interface 1998, pp. 455-461, Nagoya Japan, 1997.

[14]R. Simmons and D. Apfelbaum. "A Task Description Language for Robot Control." In Proc. Conf. on Intelligent Robotics and Systems (IROS), Vancouver Canada, 1998.

[15]K. Singh and K. Fujimura. "Map Making by Cooperating Mobile Robots". In Proc. Intl. Conf. on Robotics and Automation, 1993