Energy Efficient Multi-Robot Exploration

Toby Baker and Tarkan Al-Kazily

***Abstract*— Search and Rescue operations can often pose serious hazards to rescue personnel. Autonomous robot exploration is attractive for search and rescue operations as it can alleviate much of that risk. However, robots have limited battery capacity and the cost of production so it is ideal to keep this to a minimum. The work we present in this paper uses the concept of frontier exploration and addresses the exploration problem by creating policies to encourage multiple robots to explore in the most energy efficient manner. When navigating with two robots the total exploration time was reduced by 60% and the total cumulative distance travelled increased by only 10%.**

I. INTRODUCTION

The inspiration for this project came when looking into the DARPA subterranean challenge [1]. In Search and Rescue operations environments can change rapidly and often pose significant risk to human personnel. Mobile robots are a great solution to this problem and can augment the process by carrying out mapping and hazard identification. With lives at risk in such scenarios it's ideal for this process to be as fast as possible, while adhering to hardware limitations.

Most exploration techniques are based off the idea of frontiers [2]. Frontiers are the boundary between explored and unexplored space. By navigating to frontiers the robot can uncover more unexplored space. Thus, navigation becomes the task of navigating to frontiers, until there are no frontiers left. When there are no frontiers left, the map has been fully explored.

Frontiers give us a set of targets to navigate to, however, the next question becomes in what order to we go to each of the frontiers? In particular how can we do so in a time (to save lives) and energy efficient manner (to conserve battery). A group of researchers from Purdue University were able to reduce the energy consumed by an individual robot by 42% when compared to other, utility based exploration techniques [3].

This algorithm encourages the robot to tackle the frontiers which are near the robot, regardless of size. In doing so, they minimize duplicate coverage of map areas as well as back tracking. However, when maps are large and time is of the essence, one robot will not be sufficient. In this project, we build on the energy-efficient frontier-based exploration algorithm in [3]. Our goal is to encourage robots to search different map regions so that only one robot needs to uncover a given section of the map, while utilizing the same energy efficient search policy.

This work was not supported by any organization

T. Baker is with Electrical Engineering and Computer Sciences, University of California Berkeley toby_baker@berkeley.edu

T. Al-Kazily is with Electrical Engineering and Computer Sciences, University of California Berkeley tarkan@berkeley.edu

II. RELATED WORK

A. Single Robot Exploration

Exploration has been a well studied task in Robotics for years. For autonomous task completion in unknown environments, robots must first explore its surroundings and learn where the important features are, and how to localize itself. [2] is a foundational work in autonomous exploration by identifying the concept of frontiers - the boundary between free and unexplored space. By navigating to successive frontiers, a robot will continually expand their map of the environment until it has been fully explored. This strategy of frontier-based exploration influenced many later works, as well as our own project.

We can measure exploration algorithms by multiple criteria. The most popular is that of utility, or information gain, which measures how much knowledge is explored in the map over time. Prioritizing utility has the goal to explore as much of the environment as possible with every action. [4] details a utility-based exploration algorithm with a single robot that complements SLAM reconstruction by navigating to positions that provide as much information as possible to aid mapping. They sample candidate positions in explored space to and evaluate them for the potential new area that can be seen and penalize based on distance to the position. This algorithm allowed them to construct maps with fewer navigation actions.

Other approaches such as [5] have been able to balance utility gain with the accuracy of the map, as well as low uncertainty in the robots pose. When high uncertainty is detected in the robot's pose the selection process considers how much of the known environment it will see in order to reduce this. Such approaches are able to generate more accurate maps, while still exploring the map in an efficient manner.

Another criteria is that of energy usage. [3] addresses prioritizing robot energy usage when doing frontier-based exploration by prioritizing frontiers based on relative robot orientation over utility. In this way, the exploration algorithm features less backtracking and more efficient map coverage in terms of distance travelled, which directly corresponds to reduced energy usage. We explain this search policy in more detail in our methodology section, and how we extend it to work with multiple robots.

B. Multiple Robot Exploration

Exploration algorithms for a single robot are limited by the speed of the robot, its capabilities, and its energy storage. Changing any of these features may be costly in order to

quickly explore larger environments, but a natural approach is to use multiple robots for exploration. Coordinated strategies rely on having multiple robots communicate and share their observations to construct a resulting map in less time than a robot could complete on their own.

[6] is one of the first examples of coordinated exploration that builds on [2] by combining each robots' observations into a common map to be able to account for the explored regions of other robots. Their minimal approach does not handle cases where robots navigate to close frontiers, meaning it misses many guarantees on efficiency. [7] provides another realization of using a team of robots to explore the environment, by prioritizing utility gain and penalizing redundant coverage by multiple robots. In this way, robots coordinate themselves to spread in different directions based on their speed and location to more quickly explore the full map.

Other approaches have used a market architecture [8], [9]. In this framework robot's are negotiate with each other and share map information in order to try and better their own position. In this free market approach robots evaluate paths with revenue and cost functions. Each robot then bids for all targets to be allocated a sub-task. [8] is very useful in a practical sense since it is robust to loss of robots and can handle dynamic additions of more robots.

III. METHOD

A. System Overview

Due to circumstances outside of our control, we relied on simulation to develop our project and used STDR Simulator to model unicycle robots in a 2D map. Every mobile robotics exploration system needs a motion planner, sensing system and mapping tool. We use Move Base for motion planning due to its native compatibility with unicyle model robots, a simulated laser scan for sensing, and Gmapping to construct a map from the laser scans.

Once the environment was setup, we drew inspiration from [3], to write an energy efficient, frontier selection process for one robot. To do this we first needed to identify frontiers in a costmap which is outlined in III-B. Once all the frontiers were identified we would be able to implement and test our energy-efficient algorithm for a single robot. Finally, we then wanted to expand the systems capabilities by allowing multiple robots to coordinate their exploration in order to explore different regions of the map in a time and energy-efficient manner. We detail these algorithms in III-C.

We developed our work to easily extend to multiple robots by using a client-server model with a centralized server, exposed as a ROS Service (Figure 1). Each robot is an instance of a client, and requests a target from the server. The server incorporates robot observations using the multirobot_map_merge package [10] into a single map in order to execute our frontier identification and selection algorithms. The server's response contains the new goal for the robot, which it sends as a navigation goal to Move Base.

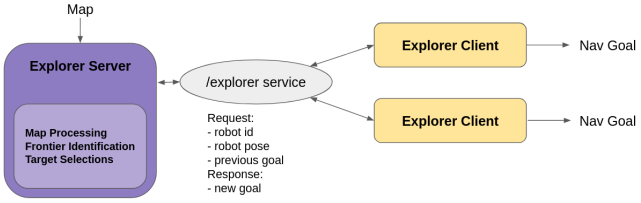


Fig. 1: Our coordinated exploration client-server model, in which robot clients request targets from the centralized server.

B. Frontier Identification

Frontier identification is the process of locating cells which are unexplored, yet have a neighbouring cell that is considered to be free space. In essence we are trying take the initial occupancy grid seen in Figure 2 and identify the coloured cells. For our project, we are making the assumption that the environment is static, and that each robot does not appear in the other robot's laser scans (a consequence of our simulator).

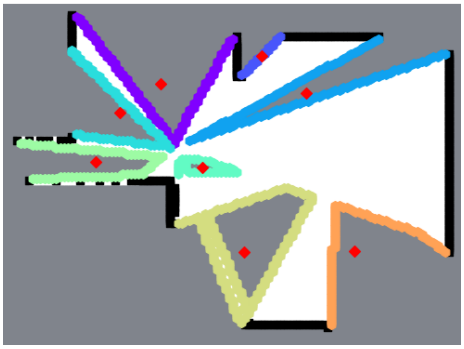


Fig. 2: An Occupancy Grid with each frontier highlighted as a different colour.

The frontier identification algorithm is outlined in Algorithm 1 and is a breadth first search on the free cells of the map. On the first iteration we will search the whole set of free space, however, after this the search begins from where we left off (i.e. the leaf nodes of the graph). This optimization makes use of our assumption that the environment is static.

The filter_frontiers function at the end of Algorithm 1 was used to remove internal frontiers as well as 'small' frontiers which do not consist of many cells. In Figure 2 the frontier in the centre would be considered internal and would be removed by the filter.

C. Frontier Selection

1) Normal Case: This section will cover the algorithm used to select energy efficient frontiers to go to for the single robot case, which was heavily inspired by [3]. The resulting algorithm can be seen in Algorithm 2, which takes in a list of frontiers and the robot's pose and returns the target the robot should go to. In Figure 3a we see that frontier 1 would be selected, since it is within the sensing radius and has the smallest angle from the robot's y-axis. The next best target

Algorithm 1: Frontier Identification Algorithm

Data: map
Result: A list of frontier data

```
start_cell = world2map((x, y)_robot0);
if start_cell != free_space then
    start_cell = nearest_free(start_cell)
queue.append(start_cell);
while queue is not empty do
    cell = queue.pop_left();
    for point in 4neighbourhood(cell) do
        if isNewFrontierCell(point) then
            frontiers.append(buildNewFrontier(point))
        else if isFreeSpace(point) and point not in
          explored_cells then
            explored_cells.append(point) ;
            queue.append(point) ;
return filter_frontiers(frontiers)
```

would be frontier 2 and finally frontier 3 since it is out of range.

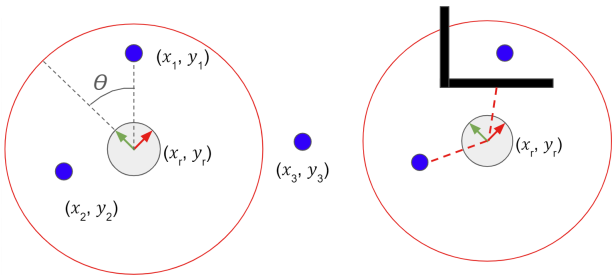
The other important part of the algorithm is considering frontiers to which the robot does not have a clear line of sight as an 'outside' frontier. This is illustrated in Figure 3b. In this instance the occluded frontier would not be selected, and the one behind the robot would as the total energy consumption to get there will be lower.

Algorithm 2: Frontier Selection Algorithm

Data: frontiers, robot_pose = r
Result: A target pose for the robot

```
within = [] # stores in range targets ;
outside = [] # stores out of range targets;
for front in frontiers do
    f = g_robot^-1 * front.pose;
    dx, dy = (f_x - r_x), (f_y - r_y);
    euclidean_dist = sqrt(dx^2 + dy^2);
    isClear = rayTraceToFrontier(front);
    if euclidean_dist < sensing_radius and isClear
      then
        angle = atan2(dx, dy);
        within.append((front, angle));
    else
        outside.append((front, dist))
if if within is not empty then
    return smallestAngle(within)
else
    return smallestDistance(outside)
```

The algorithm described above works well in the multi-robot case when the robots are not nearby each other. In other words, they are exploring their own regions. In Section III-C.2 we will cover how to handle situations where the robots



(a) Frontiers fall inside the robot sensing radius

(b) The space between the robot and the frontier is free space

Fig. 3: In-Sensing-Region Frontier Selection Criteria

are close to each other.

2) Multi-Robot Case: Figure 4 is a common occurrence, particularly on startup, which occurs when the robots are near to each other. According to the standard policy described in Section III-C.1, both robots would go to the green frontier resulting in duplicate coverage of an area, which is inefficient. In situations like this it is more ideal to spread the robots out so they can each explore their own individual map area.

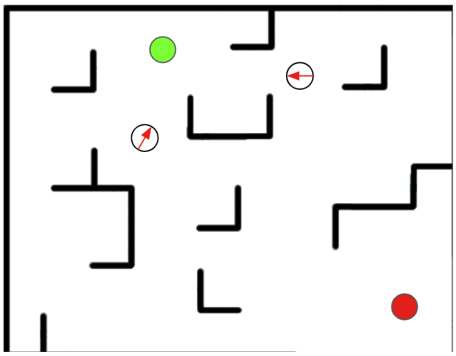


Fig. 4: Example of two nearby robots selecting the same frontier target.

A more illustrative diagram of this edge case can be seen in Figure 5. Both robots according to the standard policy would navigate to the pink frontier. Our method begins by first assigning the pink goal to one of the robots. To make a decision about where the second robot would go we first calculate the euclidean distance between the pink frontier and all other frontiers in the map. We then select the frontier that would maximise the resulting distance between the robots once they reach their goals. It is important to note that this approach generalizes well to more than two robots and simply tries to maximise the total resulting separation from all robots goals.

The overall aim of this approach is to ensure only one robot covers any given area of the map. In an ideal scenario this would cut total exploration for n robots by a factor of n, and cut the distance travelled for each robot by a factor of n.

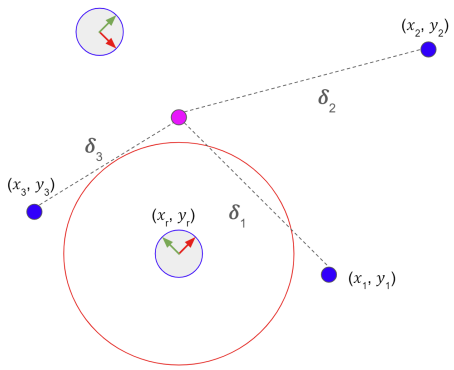


Fig. 5: To distance the robots the frontier with the furthest distance from all other robot goals is chosen

IV. RESULTS

A. Metrics

Distance Travelled: By integrating the velocity information over time we were able to calculate the distance travelled for each robot. We quantified a rotation of pi radians as 1 unit of distance and 1 m as a unit of distance.

Exploration Time: The time taken to uncover all of the frontiers in the map.

B. Experimental Setup

For our experiments we were using ROS Melodic and Ubuntu 18.04 as well as as multirobot_map_merge [10], move_base for control and planning and gmapping for SLAM.

When increasing the number of robots we were hoping to keep the total distance travelled by all robots to be similar to the total distance travelled in the single robot case. Furthermore, with the addition of n extra robots we were hoping to see close to $n\times$ reduction in total exploration time.

For the server to have a reasonable knowledge of where the robots are each client would send an update every 2 seconds.

C. Results

Algorithm	N Robots	Time (s)	Distance
Greedy [10]	1	641	160
	1	254	87
	1	408	127
Energy	1	306	102
	1	340	95
	1	370	106
Energy	2	175	49+56=105
	2	171	57+46=103
	2	218	60+63=123

We conducted three trials of exploring STDR sim's default map with our algorithm using one and two robots, as well as using the greedy exploration package explore_lite with a single robot from [10], with our measured results given in Table IV-C. In the single robot case we see that our algorithm has better averages in distance and runtime than the greedy exploration algorithm, which demonstrates that overall our method has less backtracking. We see approximately a $1.8\times$ decrease in both the distance travelled and the total exploration time when scaling the system up to two robots. We found these results to be very encouraging. To approach our goal of a $2\times$ reduction with two robots we identified several improvements that we think would get us to this goal that are outlined in V.

Furthermore, from Table IV-C we see that the system was reliable and predictable in terms of the distance and time travelled. Particularly when compared to the Greedy algorithm, which just aims to select a frontier within the sensing region. Predictability and speed are desirable in search and rescue scenarios, therefore we believe a system using such an algorithm could prove very useful and ideally the decrease in time and distance travelled would continue to increase in proportion with the number of robots. Furthermore, the cost of each robot could be kept low by requiring them to have less battery storage allowing a fleet of robots to quickly map a disaster area. Furthermore, by tuning the radius by which you consider the robots to be close you could tune the size of the region each robot would search in.

V. CONCLUSIONS AND FUTURE WORK

We were aiming to see an n fold reduction in exploration time and distance travelled for n robots. We were able to get reasonably close to this ideal goal and believe this direction of exploration is very promising. Despite this, we have identified several areas where we think the algorithm could be improved to approach this goal.

One issue that arose with our spread algorithm was that although it encouraged the robots to spread, it didn't account for energy use at all. An example of this can be seen in Figure 6. In such a scenario it's clear it would be better for this robot to clear the bottom left of the map. Including a better spread heuristic, which could include the cost for estimated distance travelled could solve this, or possibly optimizing between spread and the density of frontiers in a given region.

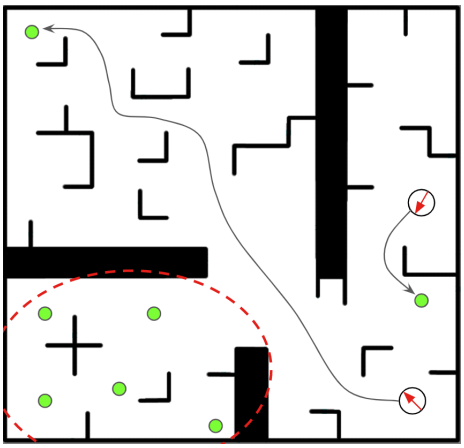


Fig. 6: To distance the robots the frontier with the furthest distance from all other robot goals is chosen

One key difference between our energy-efficient algorithm

and the one used in [3], is that we use frontier centroids rather than frontier cells when performing ray tracing and setting targets. This causes the issue seen in Figure 7. All the frontiers in this image should be considered accessible however, the red frontiers will be considered occluded despite being easily accessible. Using centroids did however, encourage the robot to stay away from walls. We believe a good compromise would be to use the cells for ray tracing and the centroids when choosing navigation goals.

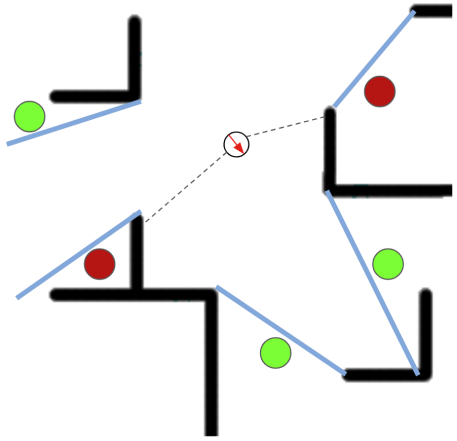


Fig. 7: To distance the robots the frontier with the furthest distance from all other robot goals is chosen

Another improvement that we think would be very useful is adapting the metric when considering frontiers outside the sensing radius. In Figure 8 we see the robot selects the closest frontier by euclidean distance, despite the fact it is clearly better to select the frontier to its right. Incorporating some kind of approximate path length cost, rather than using the euclidean distance could solve this issue.

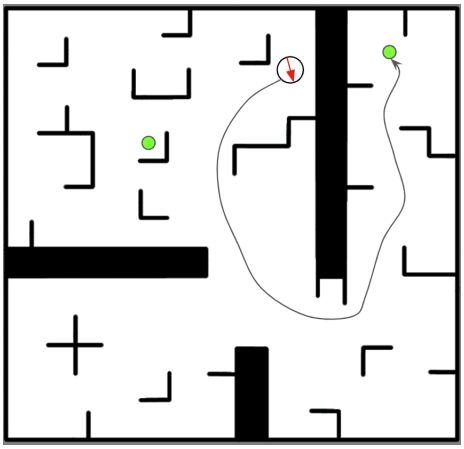


Fig. 8: To distance the robots the frontier with the furthest distance from all other robot goals is chosen

We also think it would be worthwhile to modify the way we were interacting with some of the existing packages. For example, move base has a set of recovery behaviours which result in a lot of rotating in place. By writing custom plugins one could modify the way these work to make them more energy efficient. Furthermore, we were merging maps often and this could cause delays in how long it took to update frontier locations, possible sending the robot backtracking. We thus think it would be better to periodically update the global map as in [6] when the robot reaches a new frontier and have each robot locate frontiers in it's local map.

Overall, we were pleased with the promising results we saw in a short period of time, and would really like to see how this would perform on hardware with imperfect sensing and localization.

References

[1] B. Allen, "Unearthing the subterranean environment." [Online]. Available: https://www.subtchallenge.com/

[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA97. Towards New Computational Principles for Robotics and Automation*, Jul 1997.

[3] Y. Mei, Y.-H. Lu, C. Lee, and Y. Hu, "Energy-efficient mobile robot exploration," *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006.

[4] H. H. González-Baños and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002. [Online]. Available: https://doi.org/10.1177/0278364902021010834

[5] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters," *Robotics: Science and Systems I*, Aug 2005.

[6] B. Yamauchi, "Frontier-based exploration using multiple robots," *Proceedings of the second international conference on Autonomous agents - AGENTS 98*, 1998.

[7] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[8] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, May 2002.

[9] M. B. Dias and A. T. Stentz, "A free market architecture for distributed control of a multirobot system," in *Proceedings of 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, July 2000, pp. 115–122.

[10] J. Hörner, "Map-merging for multi-robot system," Prague, 2016. [Online]. Available: https://is.cuni.cz/webapps/zzp/detail/174125/