Multi-Robot Search And Rescue Using Dynamic Voronoi Partitioning

For Distributed Systems

Jianming Han[a], Basem Shaker[a]

[a]*Department of Industrial Engineering, University of Trento, Italy.*

July, 2022

Abstract

The objective is to search an unknown environment for a specific target using cooperative robots. The robots have no previous knowledge of the environment and need to collaboratively explore the environment to find and rescue the target. We proposed a dynamic Voronoi partition-based exploration algorithm to ensure exploration. We further used two exploration strategies of picking the exploration target. The results show that our algorithm achieves better performance when more robots are deployed. The improved exploration strategy further increased the performance by a significant margin.

1. Introduction

A rescue robot is designed to search and rescue humans in areas where natural hazards and other significant disasters happen. Typically, the mission is performed in an unknown environment, where the robot needs to sense the surrounding environment and localize itself on the map. In the context of human rescue, it's absolutely more urgent to search for the target as fast as possible. Consequently, more studies in the robotics community focus on developing a coordinated multi-robot exploration system [1, 2], which combines techniques such as localization, navigation, and communication. A coordinated multi-robot system is more advantageous than a single robot system because the robots can accomplish a mission in a cooperative manner by sharing their local maps to enable faster exploration and avoid overlapping.

One major challenge in the multi-robot scenario is to divide the space properly for each robot in order to achieve higher exploration efficiency. Voronoi diagrams [3], or alternatively Voronoi partitions, have been proved to be an effective method for space partition and applied in many coordinated multi-robot exploration systems. Voronoi partition is a partition of space into regions that are close to a given set of objects (robots in our case). The centroid, or alternatively center of mass, of each region, is calculated and treated as a goal for the object in this region. Perform it repeatedly, and an optimal space partition will be achieved for all the objects. Figure 1 shows two examples of the Voronoi diagram. On the left is the Voronoi diagram of 10 randomly selected objects (dots) in a square, and the circles are the centroids of the Voronoi cells. On the right is the optimal Voronoi diagram where the objects and centroids coincide.

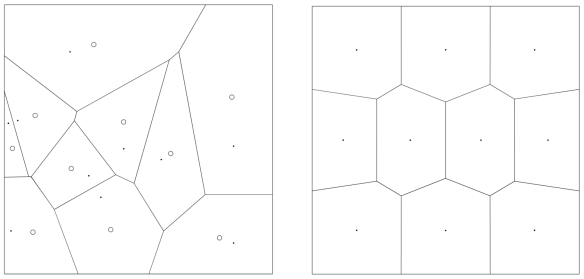


Figure 1: Voronoi diagrams. Image from [4].

The occupancy grid [5] is the most commonly used map representation for cooperative multi-robot exploration tasks [6, 7], where the continuous space is discretized as cells and each cell is categorized into *free*, *occupied*, and *unknown*. In this project, we combine the occupancy grids with the Voronoi partition for multi-robot exploration.

Many previous methods utilize Voronoi partition for multi-robot exploration tasks. In [8, 9], an initial Voronoi partition was used to divide the space for the robots. The robots then perform exploration in their specific regions. In [10, 11], dynamic Voronoi partition strategies were proposed to perform Voronoi partition repeatedly, i.e., the Voronoi partition is recomputed based on the latest global map after each information sharing. In this project, we adopt the dynamic Voronoi partition strategy because it fits into a real-world online setting.

1.1. Problem Formulation

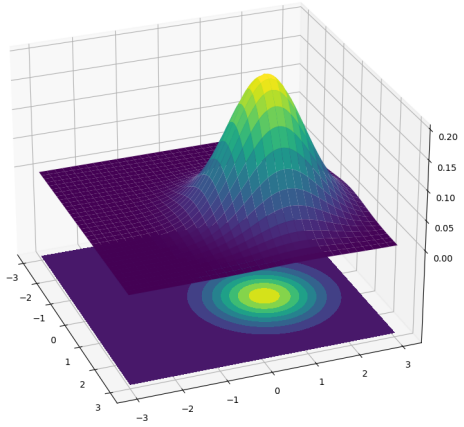


Figure 2: Example of the density function using bivariate Gaussian pdf with mean value centered in a specific point.

The objective of this project is to search for an unknown environment for a specific target with coordinated multi-robot exploration, where a dynamic Voronoi partition is adopted as the exploration strategy. In this section, we formulate the problem for this search and rescue task. We assume a limited sensing range and unlimited communication range for all the robots.

We denote the mission space as $\mathcal{Q} \in \mathbb{R}^2$, where a set of n robots are deployed. The position of i-th robot is denoted as $p_i = (x_i, y_i), \forall i \in \{1, 2, \cdots, n\}$. In a dynamic Voronoi partition setting, the Voronoi partition is performed when a new communication among the robots takes place, i.e., an update to the map merging all the sub-maps. More specifically, we perform Voronoi partition in the unknown space $\mathcal{Q}_u \subseteq \mathcal{Q}$ to ensure exploration. Thus, the i-th Voronoi cell is formulated as

$$\mathcal{V}_i = \{q \in \mathcal{Q}_u | \parallel q - p_i \parallel \leq \parallel q - p_j \parallel, \forall j \neq i\}. \quad (1)$$

Given a density function $\varphi(q)$ defined in $\mathcal{Q}$, the center of mass of i-th Voronoi cell is defined by

$$C_{\mathcal{V}_i} = \frac{\int_{\mathcal{V}_i} q\varphi(q)dq}{\int_{\mathcal{V}_i} \varphi(q)dq}. \quad (2)$$

To ensure exploration to the unknown space $\mathcal{Q}_u$, we define the density function $\varphi(q)$ to be a bivariate Gaussian pdf with mean value centered in a point $q_u \in \mathcal{Q}_u$, where

$$\varphi(q) = \frac{1}{2\pi\sqrt{|\Sigma|}} exp(\frac{-(q-q_u)^T\Sigma^{-1}(q-q_u)}{2}), \quad (3)$$

where Σ is the covariance matrix of the variances. An example of the density function is depicted in Figure 2. With such density function, the robots are forced to move towards the unknown space.

The known space $\mathcal{Q}_k$ expands during robot exploration. Assuming the target to be rescued is denoted as $p_t \in \mathcal{Q}$, we define the target is found when $p_t \in \mathcal{Q}_k$. Afterward, a new bivariate Gaussian-based density function $\varphi_t(q)$ with mean value centered in target point p_t is defined, so that all the robots move towards the target and eventually rescue the target.

2. Adopted Models

This section will discuss the type of communication system along with the system model used in the project. The coding languages used are Python and C++.

2.1. Communication System

We used Robot Operating System (ROS) as a communication system and a backbone for the whole project. ROS is a set of software libraries and tools that help build robot applications. ROS requires three programs that are necessary for its function and they are as follows:

1. ROS Master
2. Parameter Server
3. Rosout

The Master is a centralized server and uses XML/RPC protocol. XML/RPC uses HTTP as the transport and XML as the encoding. The XML/RPC system is used only to negotiate connections for data, which means data never route through the master. [12]

ROS compiles processes into Nodes that can perform computations and communicate with each other using topics, services, and actions. Each node consists of:

1. Slave API: an XML/RPC API that receives callbacks from the Master, and negotiates connections with other nodes.
2. Topic transport protocol: establishes topic connections with an agreed-on protocol. The chosen protocol depends largely on the application, however, TCP is widely used because it provides a simple, reliable communication stream.

Figure 3 shows the sequence of communication that ROS uses, which can be summarized in the following steps:

1. Publisher registers with Master asking to publish to a specific topic {XML/RPC}

2. Subscriber registers with Master asking to subscribe to a specific topic {XML/RPC}
3. Master informs Subscriber of Publisher with requested topic {XML/RPC}
4. Subscriber contacts Publisher to request a topic connection and negotiate the transport protocol {XML/RPC}
5. Publish sends Subscriber the settings for the selected transport protocol {XML/RPC}
6. Subscriber connects to Publisher using the selected transport protocol {TCP}
7. Publisher sends data to the Subscriber on TCP/IP server socket

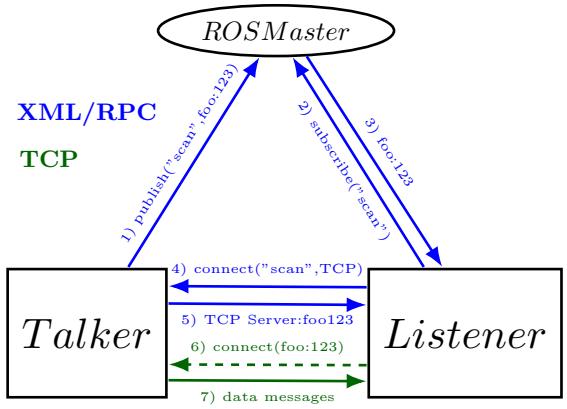


Figure 3: ROS Communication Protocol

ROS offers software that is organized in packages. A package generally contains nodes, a library, and configuration files that constitute a useful module. ROS packages provide useful functionality that can be easily reused. In our project we employed the use of the following packages:

1. gmapping: offers a simultaneous localization and mapping (SLAM) solution to calculate the current pose and generate occupancy grids for the robots.
2. multirobot_map_merge: takes the individual occupancy grids of the robots and merges them into one map.
3. move_base: provides a global and a local planner for each robot. The output of the package is a velocity control based on the goal input it is given.

2.2. System Model

Our project used Gazebo as the simulation engine. Gazebo is an open-source 3D robotics simulator that integrates ODE physics engine, OpenGL, and supports simulation for sensors and actuator control.

For the robot models, we chose Burger, the TurtleBot 3 model [Figure 4]. For motion, It has two drivable rear wheels [Dunamixel XL430 servors] and a caster in the front. It uses a Raspberry Pi as an integrated system and has a 360 Laser Distance Sensor LDS-01 2D laser scanner on top with a 1-degree resolution and 4.5 meter range.

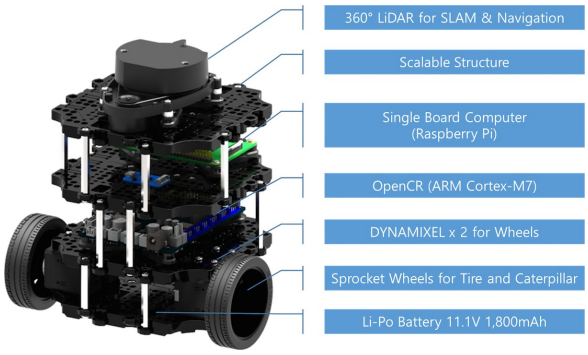


Figure 4: TurtleBot 3 - Burger [13]

3. Solution

This section provides a detailed description of the solution used for the search and rescue task.

3.1. Discretized Voronoi Partition

In our solution, we combine the Voronoi partition with the occupancy grid of the space, where we get a discretized unknown space $\mathcal{D}_u$ for Voronoi partition calculation. Thus, the Voronoi cell for robot i at each calculation is reformulated as

$$\mathcal{V}_i^d = \{q \in \mathcal{D}_u | \parallel q - p_i \parallel \leq \parallel q - p_j \parallel, \forall j \neq i\}, \quad (4)$$

where $\mathcal{V}_i^d = \{q_j\}_{j=1}^{M_i}$ and M_i is the total number of grids in its Voronoi cell. Therefore, the centers of mass is redefined as

$$C_{\mathcal{V}_i^d} = \frac{\sum_{j=1}^{M_i} q_j \varphi(q_j)}{\sum_{j=1}^{M_i} \varphi(q_j)}. \quad (5)$$

3.2. Navigation

Mapping and navigation were done using three off-the-shelf ROS packages. This section will cover their functionalities and working principles.

3.2.1. SLAM

The mutual dependency between the estimation of the robot pose and its map estimates requires searching for a solution in a high-dimensional space.

The grid mapping uses a Rao-Blackwellized particle filter. This filter applies a factorization to the SLAM posterior [shown in equation 6] and allows us to first estimate the trajectory of the robot and then compute the map given that trajectory.

Each particle in the filter carries an individual map of the environment. Once a loop closure has been achieved, the particles which have an aligned estimate will get a high updated weight, and the ones that are not would receive a low weight and eventually die out[14]

$$p(x_{0:t}, m|z_{1:t}, u_{1:t-1}) = p(m|x_{1:t}, z_{1:t}).p(x0:t|z_{1:t}, u_{1:t}) \quad (6)$$

where $x_{0:t}$ are the poses, m is the map, $z_{1:t}$ are the observations, $u_{1:t}$ are movements. The output is a probabilistic occupancy grid, where values close to 0 represent a high probability that the cell is free.

3.2.2. Map Merging

Merging the occupancy grids is essential in order for our robots to collaborate on exploring the environment. The most challenging aspect of merging is getting the transformation between the reference frames of the robots. Once the transformations are known, the maps can be overlaid on a chosen global reference frame and added together.

The package utilized uses an indirect algorithm approach to estimate the transformation by focusing on overlapping features in the maps. Those features are detected using Oriented FAST and rotated BRIEF (ORB) feature detector. The process is presented in Algorithm 1.

Algorithm 1 Map Merging [15]

Input: occupancy grids
Output: transformations from grids to refrence frame

1: **procedure** ESTIMATEGRIDTRANSFORM(*Grids*)
2: detect (ORB) features
3: find matching point pairs with RANSAC
4: estimate affine transformation
5: **end procedure**

3.2.3. Motion Planner

The Move Base contains a global and a local cost map so it can create both global and local plans. Figure 5 shows the components and the flow of messages between its blocks. The move base node takes input as a pose goal and outputs a command velocity for the robot.

The local and global cost maps build an occupancy grid based on the map and the laser scan data, then apply a user-defined inflation radius. The values on the occupancy grid denote the cost instead of occupancy. Inflation propagates cost values out from the occupied cells and decreases away with distance.

The global planner calculates a path on the occupancy grid/cost map from the current to the target pose using Dijkstra's algorithm. The local planner integrates the dynamic window approach (DWA) for collision avoidance. DWA reduces the search space only to the velocities achievable within a short time interval.

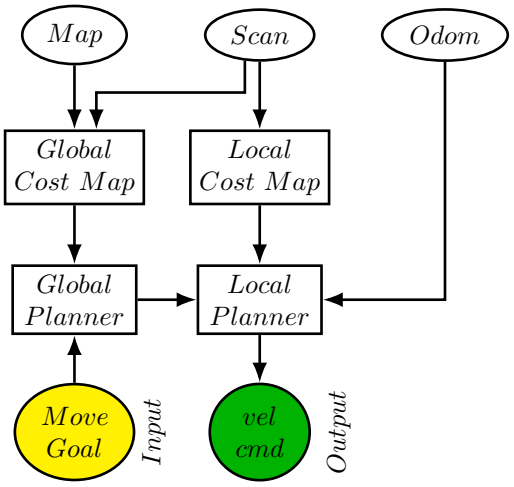


Figure 5: Move Base components

3.3. Exploration Strategy

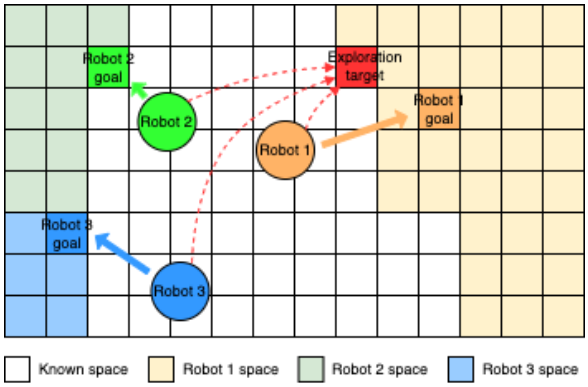


Figure 6: Illustration of our discretized dynamic Voronoi partition strategy.

We apply the discretized Voronoi partition in a dynamic setting as our exploration strategy and illustrate it in Figure 6. After each communication among the robots, we get an updated known space $\mathcal{D}_k$ and thus an updated unknown space $\mathcal{D}_u$. Then we perform Voronoi partition in $\mathcal{D}_u$ using Equation 4, which divides the unknown space into a set of Voronoi cells $\{\mathcal{V}_0^d, \mathcal{V}_1^d, \cdots, \mathcal{V}_n^d\}$. To further ensure exploration, we assume a bivariate Gaussian-based density function with its mean value (exploration target) centered at a point $q_u \in \mathcal{D}_u$. Thus, we get a set of goal positions of the robots (the centers of mass) $\{C_{\mathcal{V}_0^d}, C_{\mathcal{V}_1^d}, \cdots, C_{\mathcal{V}_n^d}\}$ following Equation 5. If $C_{\mathcal{V}_i^d} \in \mathcal{D}_k$, we pick the closest point to it from

the unknown space, thus the adjusted actual goal position of robot i subjects to

$$C'_{\mathcal{V}_i^d} = arg \min_{q_k \in \mathcal{D}_u} \parallel q_k - C_{\mathcal{V}_i^d} \parallel . \quad (7)$$

In terms of picking the exploration target q_u, we apply two strategies. In the baseline strategy, we randomly pick a point $q_u \in \mathcal{D}_u$. While in the improved strategy, we pick the closest point q_u to one of the robots from $\mathcal{D}_u$. The complete exploration strategy is summarized in Algorithm 2.

Algorithm 2 Dynamic Voronoi Partition

repeat
  robot communication $\rightarrow \mathcal{D}_k, \mathcal{D}_u$
  if rescue target $p_t \in \mathcal{D}_k$ **then**
    set p_t as exploration target
  else
    Baseline strategy:
    randomly pick exploration target $q_u \in \mathcal{D}_u$
    Improved strategy:
    randomly pick a robot i
    pick the closest point $q_u \in \mathcal{D}_u$ to robot i
  end if
  generate density function $\varphi(q)$
  for each robot $i \in \{1, 2, \cdots, n\}$ **do**
    robot position $\rightarrow p_i$
    Voronoi partition in $\mathcal{D}_u \rightarrow \mathcal{V}_i^d$
    centers of mass $\rightarrow C_{\mathcal{V}_i^d}$
    if $C_{\mathcal{V}_i^d} \in \mathcal{D}_k$ **then**
      goal adjustment $\rightarrow C'_{\mathcal{V}_i^d}$
      motion planning $(p_i, C'_{\mathcal{V}_i^d})$
    else
      motion planning $(p_i, C_{\mathcal{V}_i^d})$
    end if
  end for
  navigate to goal positions for time t
until target p_t is rescued

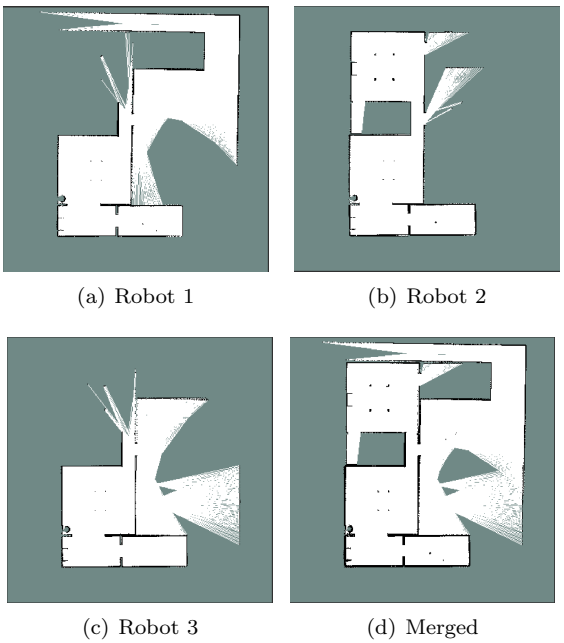
To facilitate better navigation behaviors of the robots, we further adjust the goal position to be away from high-cost areas utilizing the cost map.

4. Implementation Details

Figure 8 shows the simulation process pipeline. Gazebo generates the simulation based on environment parameters and the description of the turtlebot3 robots. Gazebo publishes odometry [pose and twist] and scans data for each robot separately on their own topics.

The gmapping package provides a laser-based simultaneous localization and mapping (SLAM) solution. The SLAM node receives the laser scan data and odometry and then outputs an occupancy grid map for each robot separately as seen in Figures 7(a), 7(b), and 7(c).

The occupancy grids are fed into a map merger node where matching features are detected to estimate the transformation and output a merged map [Figure 7(d)]. The merged map is used by both the Target Pose and the Dynamic Voronoi Partition nodes.



(a) Robot 1 (b) Robot 2

(c) Robot 3 (d) Merged

Figure 7: Occupancy grid for each Burger Robot before and after merging

The Target Pose node contains the location of the target on the map. It only publishes the pose of the target once its location has been discovered on the merged map. Once published, the robots can receive the pose and switch to the secondary bivariate Gaussian-based density function so all the robots move towards and rescue the target.

There are separate cost maps and planers for each robot that are used by the Voronoi Partition and Move Base nodes. Once The Voronoi Partition node calculates a target, it publishes it for each robot separately on the Move Base Goal topic. The Move Base node calculates the global plan based on the cost maps and executes the local plan by publishing velocity commands.

5. Results

This section showcases some selected outcomes of the project. Multiple tests have been performed and the results are shown below.

5.1. Number of Robots

The number of robots in the fleet was chosen to be three. However, we were able to also test the

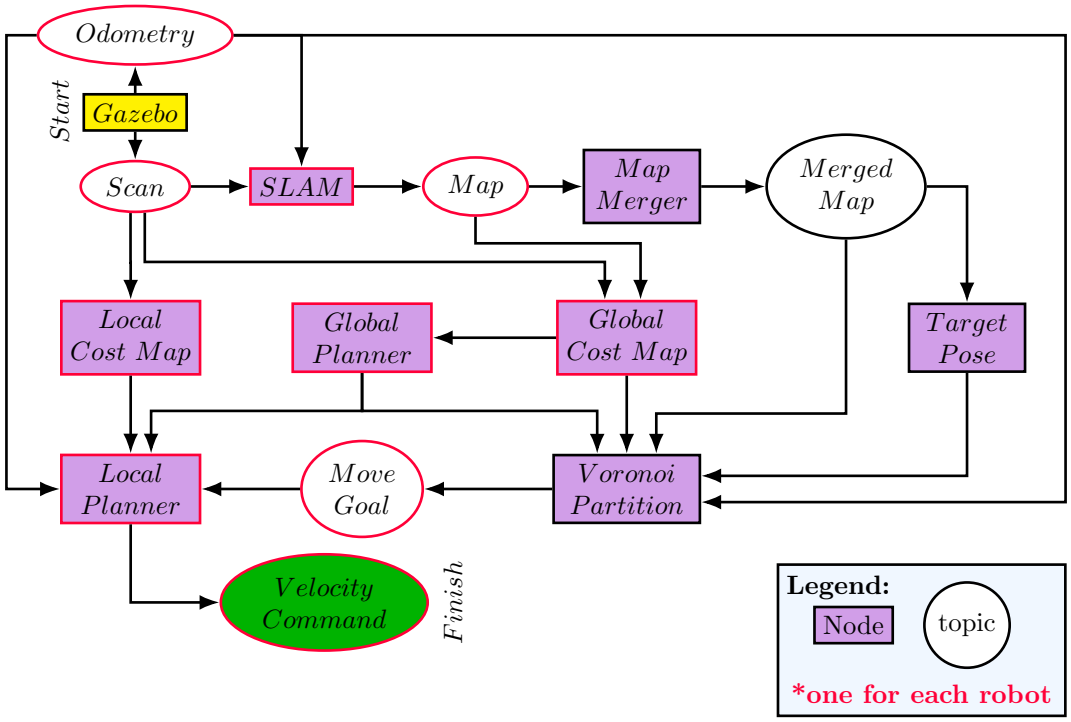


Figure 8: Process Pipeline

efficiency of using one and two robots as well. The metric we are using to compare is the time taken to discover the target. A timekeeper node starts the time counter as soon as the first move goal is published, and it stops the counter once the target location has been found.

Figure 9 is a box plot of the time taken to find a target to compare using one vs. two vs. three robots with the baseline strategy. The data includes 30 runs per number of robots. The results confirmed what was expected and justified the use of extra robots for the rescue as the time taken improved by 51% from using one robot vs three.

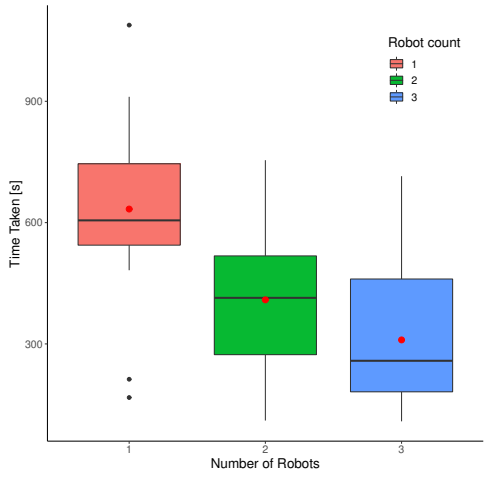


Figure 9: Time taken based on number of robots used

The discovery rate metric was calculated using the percentage of the map that has been discovered at the moment of finding the target with the baseline strategy [Figure 10]. Having three robots yields a discovery rate of 0.30% of the total map per second while using only one robot is 0.16%.

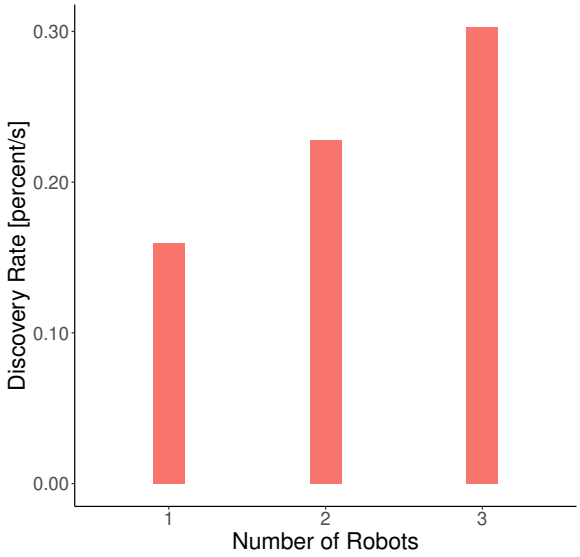


Figure 10: Discovery rate based on number of robots

5.2. Comparison of Strategies

We compared the two proposed strategies for picking the exploration target in the unknown space.

In the baseline strategy, we pick the exploration target randomly, while in the improved strategy, we pick the closest point to a robot from the unknown space. The comparison for the case of 3 robots is shown in Figure 11. We can observe that the improved strategy achieves an improvement of 46.0% for the search time compared to the baseline strategy. Moreover, the improved strategy has much lower variance which indicates that it's more stable than the baseline strategy.

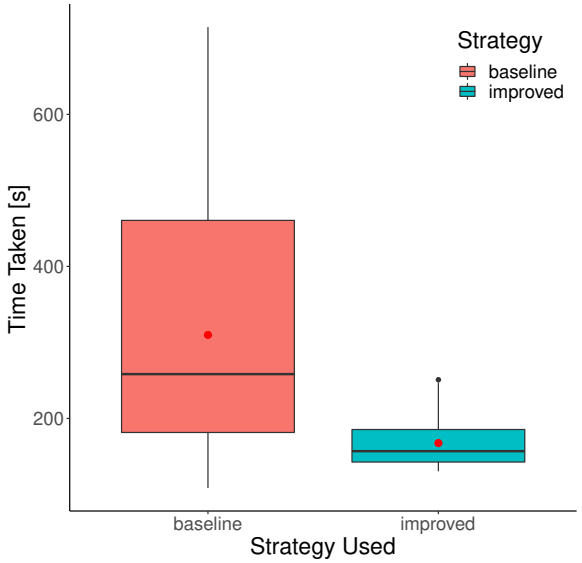


Figure 11: Results of the improved method with three robots

The results of the experiments mentioned above are summarized in Table 1.

Robot count	Time taken [s]	Percent improvement
1	633	-
2	408	35.5 %
3	309	51.1 %
3 (improved)	167	73.6 %

Table 1: Robot count comparison results

5.3. Voronoi Partition

Figure 12 shows the dynamic Voronoi partition at different iterations for a setting of three robots, where the robots are denoted as circles and the computed centers of mass are denoted as stars. An example of goal adjustment can be found at iteration 8 where one center of mass is located in the known space which is further adjusted by picking the closest point in the unknown space.

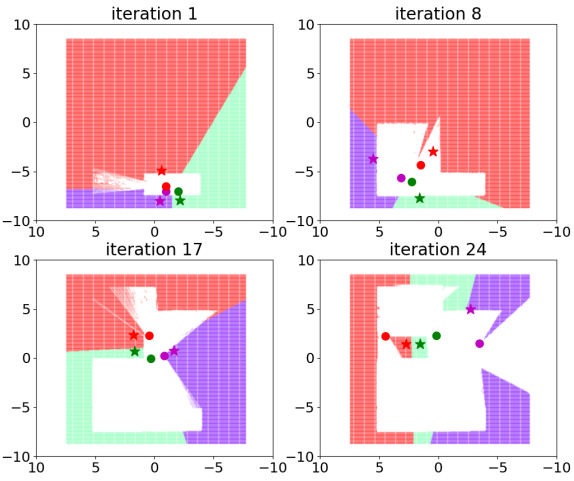


Figure 12: Results of the improved method with three robots

We further applied the coverage cost calculation in the unknown space and compared the cases of 2 robots and 3 robots. We performed 10 runs for each scenario and the results are depicted in Figure 13. It is clear to see that the 3 robots scenario has a smaller cost than the 2 robots scenario throughout the exploration process.

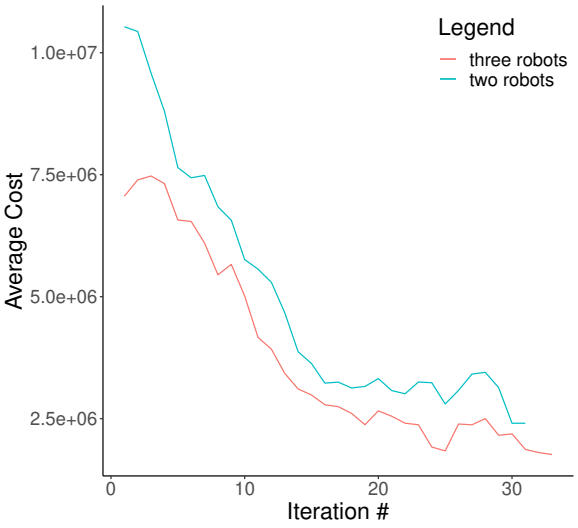


Figure 13: The cost during the iterations with three robots vs. two robots [10 runs]

6. Conclusion

In this project, we investigated the coordinated multi-robot search and rescue problem in an unknown environment. We proposed a dynamic Voronoi partition algorithm applied in the unknown space to ensure exploration, where two strategies

of picking the exploration target were further proposed and compared. The experiments show that our algorithm performs better when more robots are deployed for the search and rescue task (improvement of 51.1% for 3 robots compared to 1 robot). Further experiments verify that the improved exploration strategy improves performance by a significant margin (46.0%). For future work, we will design different maps to verify the generalization ability of our proposed algorithm.

References

[1]L. Macedo and A. Cardoso, ≪Exploration of unknown environments with motivational agents≫, in Proceedings of the third international joint conference on autonomous agents and multiagent systems-volume 1 (2004), pp. 328–335.

[2]D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, ≪Distributed multirobot exploration and mapping≫, Proceedings of the IEEE **94**, 1325–1339 (2006).

[3]F. Aurenhammer, ≪Voronoi diagrams—a survey of a fundamental geometric data structure≫, ACM Computing Surveys (CSUR) **23**, 345–405 (1991).

[4]Q. Du, V. Faber, and M. Gunzburger, ≪Centroidal voronoi tessellations: applications and algorithms≫, SIAM review **41**, 637–676 (1999).

[5]A. Elfes, ≪Using occupancy grids for mobile robot perception and navigation≫, Computer **22**, 46–57 (1989).

[6]W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, ≪Collaborative multi-robot exploration≫, in Proceedings 2000 icra. millennium conference. ieee international conference on robotics and automation. symposia proceedings (cat. no. 00ch37065), Vol. 1 (IEEE, 2000), pp. 476–481.

[7]C. Stachniss and W. Burgard, ≪Exploring unknown environments with mobile robots using coverage maps≫, in Ijcai, Vol. 2003 (2003), pp. 1127–1134.

[8]T. Patten, R. Fitch, and S. Sukkarieh, ≪Large-scale near-optimal decentralised information gathering with multiple mobile robots≫, in Proceedings of australasian conference on robotics and automation, 0.1 (2013), pp. 1–15.

[9]M. Schwager, M. P. Vitus, S. Powers, D. Rus, and C. J. Tomlin, ≪Robust adaptive coverage control for robotic sensor networks≫, IEEE Transactions on Control of Network Systems **4**, 462–476 (2015).

[10]S. Kemna, J. G. Rogers, C. Nieto-Granda, S. Young, and G. S. Sukhatme, ≪Multi-robot coordination through dynamic voronoi partitioning for informative adaptive sampling in communication-constrained environments≫, in 2017 ieee international conference on robotics and automation (icra) (IEEE, 2017), pp. 2124–2130.

[11]J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, ≪Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning≫, IEEE Transactions on Vehicular Technology **69**, 14413–14423 (2020).

[12]*Ros technical overview*.

[13]R. Advance, *Turtlebot3 burger*.

[14]G. Grisetti, C. Stachniss, and W. Burgard, ≪Improved techniques for grid mapping with rao-blackwellized particle filters≫, Robotics, IEEE Transactions on **23**, 34–46 (2007).

[15]J. Hörner, ≪Map-merging for multi-robot system≫, in (2016).