2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

A Distributed Algorithm for Mapping the Graphical Structure of Complex Environments with a Swarm of Robots*

Adam Caccavale[1] and Mac Schwager[2]

***Abstract*— This paper presents a novel multi-robot mapping algorithm which allows a large number of simple robots to map the discrete graphical structure underlying an environment of multiple disjoint subregions. Examples of such environments include rooms in a building, buildings in a town, chambers in a cave network, or islands in an archipelago. Each robot is limited to a small communication range, compass, GPS sensor, and a short-range proximity sensor (e.g. bump sensor). Furthermore memory is limited, so no metric map of the environment is stored. Instead the algorithm determines which robots inhabit the same subregion, and which of these groups of robots are able to communicate. This information is captured in a disk graph representation. It is proven that these simple capabilities are sufficient to guarantee that all agents will determine the graphical structure in a finite time. Two environment configurations were tested with a range of quantities of robots. These simulations confirm that processing time is polynomial in the number of robots and indicate that the number of steps to convergence is linear in the number of robots.**

I. INTRODUCTION

In this paper we present a scalable, decentralized control strategy by which a large number of simple robots can map the discrete structure of an environment consisting of multiple disjoint subregions. We use the metaphor of islands in an archipelago throughout the paper due to the convenience of the terminology associated (e.g. islands for disjoint regions and water for obstacles). The algorithm can be applied to exploration and topological mapping of a variety of environments, including rooms in a building, buildings in a town or city, chambers in a network of caves, and natural environments with trees, rocks, or other features that constrain mobility of robots to disjoint subregions. An example environment along with its corresponding graphical representation are shown in Figure 1.

Initially, each robot only knows of its own existence, but has no knowledge of the geometry of its island, other islands, or the topology of how these islands are located with respect to one another. Using our algorithm, each robot first discovers the existence of other robots on its own island, and on one-hop neighbor islands. It then uses communication with these discovered robots to determine the connections between one hop islands. As this information gets filled in, the robots discover the existence and location of other robots that are multiple hops away through communication. The motion, robot discovery, and communication all proceeds simultaneously, and over time all robot converge to the correct graphical representation of the environment. The key to this approach is to rely on the flow of information between robots, allowing them to act as relays for information. Furthermore by using simple control algorithms in a finite state machine architecture, little about the environment needs to be recorded, removing the need to store a map in memory.

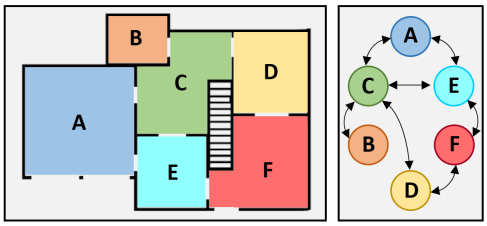


Fig. 1: Building floor plan divided into rooms and its corresponding graphical representation.

The algorithm makes use of a stigmergy principle to classify neighbors on its own island vs those off its own island. When communication is established, a virtual "checkpoint" is dropped at the location where the neighbor was sensed. It then tries to reach that virtual checkpoint. If it can get there, the neighbor must have been on its own island, otherwise it is on another island. The algorithm also takes advantage of the fact that, lacking any information about the robot's environment, traversing the island perimeter provides an advantageous path for discovering neighboring robots. Additionally knowing that the robots will eventually end up circling the perimeter results in unique steady state route that can be leveraged to make guarantees about meetings between robots on neighboring islands.

More specifically, the contributions of this paper are as follows. We present a distributed algorithm for a group of robots, with no initial knowledge of an environment, to determine the graphical structure of the environment. We prove that, using the algorithm, all robots obtain the true graphical representation in finite time. Several simulations are run to validate its efficacy. These simulations demonstrate a polynomial trend in processing time vs. number of robots as well as a linear trend in number of processing cycles vs. number of robots.

A. Related Work

The proposed algorithm builds upon and uses several existing techniques to map the graphical structure of the unknown environment. [1] and [2] use very simple sensors,

*This research was supported in part by NSF grant CMMI-1562335 and ONR grant N00014-12-1-1000. We are grateful for this support.

[1]A. Caccavale is with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 awc11@stanford.edu

[2]M. Schwager is with the Department of Aeronautics & Astronautics, Stanford University, Stanford, CA 94305 schwager@stanford.edu

along with a disk-graph representation of the environment. However in this problem formulation the graph is already known and the sensors are static. The algorithm is then able to track mobile agents moving throughout the environment. [3] uses a distributed approach for exploring an area where small graphs of robots form in the larger swarm. However, the focus for that paper is on building a metric map rather than seeking a graphical environment representation. An important aspect of these algorithms is that the robots converge on the same correct solution. This is related at a high level to the well-studied area of consensus, as exemplified by [4].

Similar to the proposed algorithm in this paper, a communication-based finite state machine is used for control of distributed robots in [5]. However, the goal of that paper focuses on the coverage of an environment as opposed to determining the graphical structure of the environment.

A key part of the algorithm presented in this paper determines if a robot is on the same island by trying to reach its last-known location. This method is based around the bug algorithm which is explained in detail in [6] and [7]. This algorithm provides a simple path planning method using limited sensors similar to the capabilities of the robots in this paper. We will show that this algorithm results in information flowing throughout the network of robots. A similar result could have been achieved using flooding, a network communication technique (see [8] and [9]). Flooding differs from the approach used in the presented algorithm, but the core idea of information defusing through the network is important.

There are many methods for distributed exploration of an environment. [10] develops a distributed clearing algorithm for sweeping through areas while guaranteeing that no intruder is missed. [11] uses distributed, multi-robot SLAM to map the environment and [12] focuses on using a distributed bidding process to solve the coordination challenge of multi-robot exploration. The perimeter tracking methods discussed in [13] and [14] are also similar to the steady-state reached by the robots when following the proposed algorithm. While the goal of these exploratory algorithms vary, none focus on the graphical structure of the environment, and none use minimalistic robots as we do here.

[15] proposes an algorithm to co-localize simple robots in a distributed fashion. Their goal of achieving their task with minimal robot capabilities is similar to our proposed algorithm, and the distributed co-localization technique could be incorporated into future versions of our algorithm to further reduce the number of sensors required.

The rest of this paper is organized as follows. We define the parameters and characteristics of the environment and robots and formally state the problem in Sec. II. In Sec. III we present our multi-robot control strategy and analyze its completeness. Finally, in Sec. IV we show numerical simulations for two environment layouts with numbers of robots ranging from 20 to 100, and in Sec. V we give our conclusions.

II. Problem Formulation

We consider the problem where a set of robots, with indices $r \in R$ and $R = \{1, \ldots, n\}$, are randomly distributed throughout a bounded archipelago-like environment, $E \subset \mathbb{R}^2$. The environment is divided into m disjoint subregions (called islands), $E = \bigcup_{i=1}^{m} E_i$ and $E_i \cap E_j = \emptyset$, separated by a pervasive obstacle E^C (called water). Each island E_i is connected, but may be non-convex, and may itself surround obstacles (called lakes). The structure of this environment can be represented as a disk graph, $G_{true} = \{L_{true}, \mathcal{E}_{true}\}$. Islands make up the nodes of the graph. If two islands are within a certain distance of one another they are considered connected and there is an edge (see the graph depicted in Figure 2 for an example).

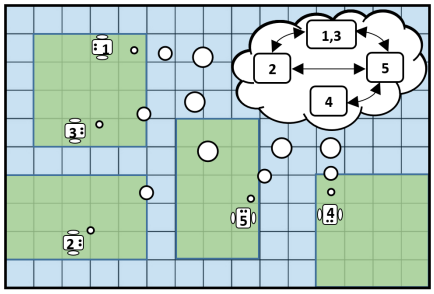


Fig. 2: Example environment with graphical representation for 5 robots and 4 islands. All robots have discovered the island's graphical structure.

Instead of representing the islands by an index, each island is represented by the ID number of the inhabiting robots which is referred to as the island name. The set of islands names, L_{true}, is a subset of all possible combinations of robot IDs, $N = \{\nu \mid \nu \subset R\}$ and $L_{true} \subset N = \{\nu_1, \nu_2, ..., \nu_n\}$. The Gromov–Hausdorff distance between two islands is $d_{ij} = dist(E_i, E_j) = \min_{\substack{\forall e_i \in E_i \\ \forall e_j \in E_j}} \|e_i - e_j\|$. When this distance is less than the robot communication range, $d_{ij} \leq D_{com}$, an edge ($\mathcal{E}_{ij} = \{\nu_i, \nu_j\}$) exists and the islands are referred to as connected. The edges of the graph are represented by an adjacency matrix $A = [a_{ij}]$. If the islands are connected $a_{ij} = 1$ and $a_{ij} = 0$ otherwise.

As robot r discovers other robots, or neighbors, it builds an estimate of the graph, $G_r^t = \{L_r^t, \mathcal{E}_r^t\}$. This observed graph evolves over time, $t = 1, 2, \ldots$, and our objective is for all robots to discover the true underlying graphical structure of their environment ($G_r^t \to G_{true}$ as $t \to \infty$ for all r). For example, the r^{th} robot may have observed robots 1, 6, and 8 on one island, robot 2 on a second island, and robot 3 on a third island. Therefore this robot's set of island names would be $L_r^t = \left\{\nu_1^{r,t}, \nu_2^{r,t}, \nu_3^{r,t}\right\}$ where $\nu_1^{r,t} = \{1, 6, 8\}$, $\nu_2^{r,t} = \{2\}$, and $\nu_3^{r,t} = \{3\}$. It might later discover, however, that the true graph structure is actually $L_{true} = \{\nu_1, \nu_2\}$ where $\nu_1 = \{1, 6, 8\}$, $\nu_2 = \{2, 3\}$. Only graphs that are connected are considered, which by definition means any island can be reached from any other island by moving along the graph

edges. Henceforth the time superscripts will be dropped for notational convenience unless they are needed for clarity.

For convenience the environment will be discretized into an occupancy grid in order to avoid dealing with the low-level control of the robots. The position of a given robot, $p_r \in \mathbb{Z}_+^2$, is modeled as $p_r^{t+1} = p_r^t + u_j$, where $u_r \in \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\}$ s.t. $p_r^{t+1} \in E$. An example environment and corresponding graph structure are shown in Figure 2.

Each robot has restricted sensing capabilities; it can only check adjacent squares for obstacles (i.e. water) and has a limited communication radius, D_{com}, in which it can communicate with other robots. Each robot is able to use a GPS sensor to query its current position and a compass to detect its orientation. However, memory is limited and the robots do not "remember" a metric map of their environment, instead they maintain cached information regarding other robots. This allows the memory requirements for the robot to be independent of its environment and scale linearly with the number of robots. For simplicity, all robots move at a constant speed arbitrarily chosen to be 1 square per time step.

When robot s is within communication range of robot r, it receives from r the following information:

Communication Bundle.

i. The robot's ID and location, r and p_r.
ii. The robot's observed graph information, G_r.
iii. A list of the neighbors of robot r, along with the location at which they were sensed.

Since the robots cannot sense any unique features about the islands themselves, they classify the islands based on what robots are on the islands. In order to ensure comparability of island names between robots, each island name is simply the sorted array of ID numbers of the robots that are on the island.

Problem Statement. *We seek an algorithm whereby all agents will discover the true graph in finite time, i.e. $\exists t^* \ s.t. \ G_r^t = G_{true}, \forall t \geq t^*$.*

In this paper we present an algorithm, Algorithm 1, which solves this problem given the sensing, moving, and communication capabilities described in this section.

Assumptions.

i. There are at least two islands.
ii. Island perimeter lengths are not exact integer multiples of one another.
iii. All islands are inhabited by at least one robot.
iv. All sensing and communication is deterministic and error-free

Assumption i is needed to ensure that all robots on the *same* island discover one another (strangely, this cannot be guaranteed unless there are at least two islands). Assumption ii ensures that robots on neighboring islands communicate regularly, and Assumption iii precludes the possibility of an undiscoverable island. Finally, Assumption iv is important for simplifying the proof of algorithm completeness. In fact, the algorithm is still effective with stochastic sensing and communication. Relaxing this assumption and quantifying the loss in efficiency is a problem left for future work.

III. Algorithm

This section describes the algorithm that will be run by every robot in a distributed manner.

A. Algorithm Description

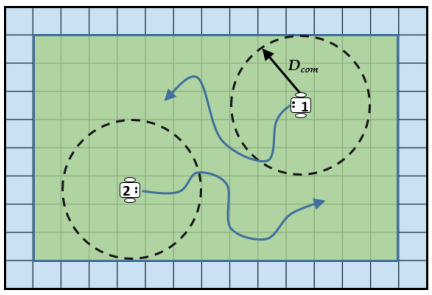
Each robot will randomly move around the island until encountering other robots or water. When other robots are located the robot will identify if the sensed neighbor is on the same island by attempting to move to the position where it sensed the neighbor. When all observed neighbors have been classified in this way, the robot moves until it finds the island perimeter, which it then follows.

1) Communication and Moving: Before moving each robot will scan for new robots within its communication range. All sensed robots, or neighbors, are then classified into one of three categories: neighbors who share the same island (N_{on}), neighbors who are on a different island (N_{off}), and neighbors who have been observed but not yet classified (i.e. unknown neighbors, N_u). Robots enter one of three modes depending on their sensed neighbors, which is used to determine how to move per Algorithm 5.

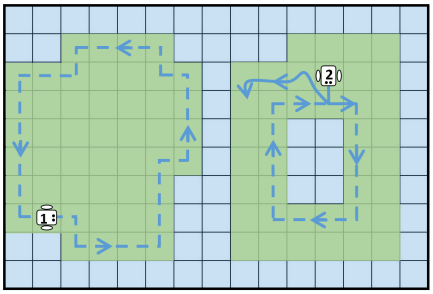
If a robot has no unknown neighbors, $N_u = \emptyset$, it will enter into one of two modes. If the robot is not adjacent to water, it defaults to *random walk mode* as shown in Figure 3a. If instead the robot is adjacent to water it will enter *perimeter-following mode* where it will keep moving such that there is always water on its right side. The robot can distinguish the perimeter of the island from that of a lake (i.e. an obstacle enclosed by an island) by tracking the integral of its change in angle from when it begins to move along the perimeter to when it returns to its original location. Since the robot always keeps the water on its right side, if the net change in angle is $< 0°$ then the path it followed must be the perimeter of a lake. In this situation the robot will then stop following the perimeter and will return to *random walk mode*. Otherwise the path it followed was the perimeter of the island and it will continue to follow this path. This behavior is illustrated in Figure 3b. [1]

If the robot has neighbors that it has not yet classified, $N_u \neq \emptyset$, it will enter *checkpoint mode*. In this mode the robot will follow Algorithm 2 to attempt to reach the position at which the first unknown neighbor was sensed. This method follows a modified version of the bug algorithm which is described in detail in [7]. Here the robot will follow the unique line connecting its initial location to the position where it sensed its neighbor, referred to as the start–goal line. If it encounters an obstacle it will follow the perimeter of the obstacle until it is back on the start–goal line and is closer to

[1] We arbitrarily chose to keep the water on the right. Left could have been chosen as well which would simply reverse this result (i.e. positive net change in angle indicates a lake).



(a) Robot moving in random walk mode.



(b) Robots moving in perimeter-following mode.

Fig. 3: Robot behaviors when all neighbors have been classified (i.e. $N_u = \emptyset$).

the goal location. Note that the robot is seeking the location where it sensed the neighbor, not the current location of the neighbor. If the robot reaches this goal location it knows it is on the same island as the neighbor and it can add the neighbor to N_{on}. If the robot returns to the location where it first encountered the obstacle it knows it cannot reach the goal, and can then add the neighbor to N_{off}. A property of this algorithm is that one of these two situations is guaranteed to occur, and there is no situation where the robot cannot be classified [7]. The newly classified neighbor's data is then merged into the robot's own information as described below. Along the way if other robots are sensed then they are added to the list of unclassified robots, N_u, and their data is cached until they can be classified.

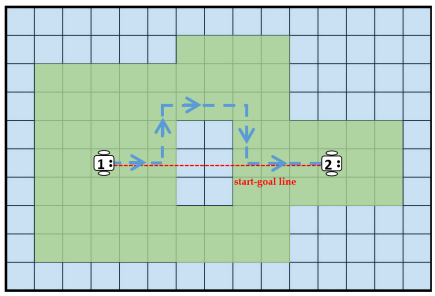


Fig. 4: Robot moving in checkpoint mode.

The most complex part of the algorithm deals with the merging of the robots' data which is the focus of the next section.

2) Island Name Merging: The merging of the robot data occurs in multiple steps. The first step is to determine what the merged island names (i.e. nodes on the graph) will be. Each robot has its set of island names which is the result of its past interactions. First, following the pseudocode of Algorithm 3, both lists of island names are combined into one large list. Then, each island name is compared against the other island names to check if there are any inhabiting robots in common, which indicates that the two islands are in fact the same island. The union of these two sets become the new island name. This is repeated until no island names have any robot IDs in common. A superscript of "-" indicates the variable has yet to be merged and a superscript of "+" means the variable has been merged. A clarifying example of Algorithm 3 is shown below.

$$L_r^- = \overbrace{\left\{\underbrace{(1,3)}_{\nu_1^{r-}}\ \underbrace{(5)}_{\nu_2^{r-}}\ \underbrace{(6,9)}_{\nu_3^{r-}}\right\}}^{\text{Set of observed islands by robot r}}, \quad L_s^- = \overbrace{\left\{\underbrace{(2,5,6)}_{\nu_1^{s-}}\ \underbrace{(9)}_{\nu_2^{s-}}\right\}}^{\text{Set of observed islands by robot s}}$$

First, append two sets of island names.

$$L^+ = \overbrace{\{(1,3)\ (5)\ (6,9)\ (2,5,6)\ (9)\}}^{\text{Appended set of observed islands}}$$

Next, continue to merge island names containing the same robot ID in L^+ until all island names are unique. In this example the ν_1^{r-} has no robot IDs in common with any other island name and will become ν_1^+. On the other hand, ν_2^{r-}, ν_3^{r-}, ν_1^{s-}, and ν_2^{s-} all have robot IDs in common and therefore will merge into ν_2^+.

$$L^+ = \overbrace{\left\{\underbrace{(1,3)}_{\nu_1^+}\ \underbrace{(2,5,6,9)}_{\nu_2^+}\right\}}^{\text{Set of observed islands after merge}}$$

3) Island Adjacency Matrix Merging: Next, the adjacency matrices are merged using Algorithm 4. The list of island names is known from the previous steps, so the size of the adjacency matrix is also known. Each element of the merged adjacency matrix can then be determined by checking if there was an edge in either of the two adjacency matrices (i.e. a_{ij}^1 or a_{ij}^2) from before the merge, in which case an edge is added in the merged adjacency matrix, $\mathcal{E}_r^+$.

B. Completeness of Algorithm

This section will demonstrate that the described algorithm is guaranteed to satisfy the given problem statement (i.e. all robots will discover the true graph and the list of robots on each island in finite time).

It is helpful to define an integer-valued measure of the difference between two observed lists of robot names, L_r and L_s. This can be thought of as the distance between two sets of names, $d(L_r, L_s) \geq 0$, and refers to the number of steps required to transform one set of names into the other where a step corresponds to one of the following actions.

i. Add a new robot ID as a new island.
ii. Merge two groups of robot IDs (e.g. ν_i and ν_j).

Algorithm 1 Main

for All Robots, r **do**
  Discover robots in communication range, N_r
  for $s \in N_r$ **do**
    if $s \in N_{on}$ *or* N_{off} **then**
      Merge Data (Algorithms 3 and 4)
    else if $s \in N_u$ **then**
      Ignore neighbor
    else
      Add robot to N_u
    end if
  end for
  Get set of adjacent obstacles, P_{obs}
  Move (Algo 5)
end for

Algorithm 2 Checkpoint Bug

while not at goal **do**
  Move towards goal along start–goal line
  if desired next step is an obstacle **then**
    Keeping the obstacle on the right, move until you encounter the start–goal line again closer to the goal and are able to move towards the goal
  end if
  if at goal location OR back at start location **then**
    Classify neighbor and exit *checkpoint mode*
  end if
end while

Let L_r^- and L_s^- be two sets of island names before merging and L_r^+ and L_s^+ be the resulting sets[1]. Note that the resulting sets are always equal, $d(L_r^+, L_s^+) = 0$. Furthermore the distance between two sets is 0 if and only if the sets are the same, $d(L_r^-, L_s^-) = 0 \iff L_r^- = L_s^-$.

Lemma 1 (Island Name Monotonicity)**.** *Given two robots with sets of island names L_r^-, L_s^-, Algorithm 3 will result in $L^+ = L_r^+ = L_s^+$ such that $d(L^+, L_{true}) \leq d(L_r^-, L_{true})$ and $d(L^+, L_{true}) \leq d(L_s^-, L_{true})$. Furthermore, at least one of the inequalities is strict if and only if $L_r^- \neq L_s^-$.*

Proof. Given two arbitrary sets of island names L_r^- and L_s^-, let $d_r = d(L_r^-, L_{true})$ and $d_s = d(L_s^-, L_{true})$. The only new information that can be brought to L_r^- by L_s^- and vice versa is knowledge of a previously unknown neighbor, a previously unknown shared island between two neighbors, or a combination of the above. Incorporating this knowledge can only reduce the number of steps between the unmerged set and the true set, $d(L_r^-, L_{true}) \leq d(L_r^+, L_{true})$ and $d(L_s^-, L_{true}) \leq d(L_s^+, L_{true})$.

Suppose $L_r^- = L_s^-$ then $L_r^+ = L_s^+ = L_r^- = L_s^-$ by the properties of Algorithm 3. Therefore $d(L_r^-, L_{true}) = d(L_r^+, L_{true})$ and $d(L_s^-, L_{true}) = d(L_s^+, L_{true})$ hence no inequality is strict. Therefore if either inequality is strict

[1]For all following variables the $^-$ and $^+$ superscripts indicate values before and after robots merge knowledge respectively

Algorithm 3 Merge Island Names

$L^+ \leftarrow$ append(L_r^-, L_s^-)
prevLength = 0
while length$(L^+) \neq$ prevLength **do**
  prevLength = length(L^+)
  for each Island Name $\in L^+$, ν_{focus} **do**
    for each Island Name $\in L^+$, $\nu_{compare}$ **do**
      if $\nu_{focus} \cap \nu_{compare} \neq \emptyset$ **then**
        $\nu_{focus} \leftarrow \nu_{focus} \cup \nu_{compare}$
        $L^+ \leftarrow L^+ \setminus \nu_{compare}$
      end if
    end for
  end for
end while

Algorithm 4 Merge Adjacency Matrices

$A^+ \leftarrow$ empty matrix of dimension length(L^+)
for ith Island Name $\in L^+$ **do**
  for jth Island Name $\in L^+$ **do**
    for each ID $\in$ ith name in L^+, r_1 **do**
      for each ID $\in$ jth name in L^+, r_2 **do**
        $ind_1 \leftarrow$ getIndexOf(r_1)
        $ind_2 \leftarrow$ getIndexOf(r_2)
        if $A_1^-(ind_1, ind2) = 1$ or $A_2^-(ind_1, ind_2) = 1$ **then**
          $A^+(i,j) \leftarrow 1$
          $A^+(j,i) \leftarrow 1$
          BREAK
        end if
      end for
    end for
  end for
end for

Algorithm 5 Move

if $N_u = \emptyset$ **then**
  if $P_{obs} = \emptyset$ **then**
    Random walk one step (mode: *random walk mode*)
  else
    Move one step keeping the obstacle on the right side
    Integrate the change in angle (mode: *perimeter-following mode*)
  end if
else
  Move one step according to Algorithm 2 (*checkpoint mode*)
end if

$L_r^- = L_s^-$ cannot be true, which demonstrates the "only if" in the final statement of the lemma.

Finally, we prove the "if" in the final statement of the lemma. In order to reach a contradiction, suppose $L_r^- \neq L_s^-$ and neither inequality is strict ($d(L^+, L_{true}) = d(L_r^-, L_{true})$ and $d(L^+, L_{true}) = d(L_s^-, L_{true})$). However, if $L_r^- \neq L_s^-$ then $d(L_r^-, L_s^-)$ must be greater than zero by the construction of the graph distance measure, i.e. $d(L_r^-, L_s^-) = 0 \iff L_r^- = L_s^-$. This is a contradiction, therefore if $L_r^- \neq L_s^-$ then at least one of the inequalities must be strict. □

Similarly, an integer-valued connectivity measure can be defined for the information captured by a robot's adjacency matrix. Let $\mathcal{E}_r$ be the set of observed island connections for an arbitrary robot with a corresponding set of island names L_r. Each element in L_r, ν_k^r, is a set of robot IDs and each island connection, or element in $\mathcal{E}_r$, specifies if there is an observed connection between these two sets. Let two robots be considered connected if they are on islands that are connected (robots on the same island are considered connected). Given two islands, E_i and E_j, the number of connections between robots is $h_{ij} = |\nu_i|\,|\nu_j|$ for $i \neq j$. For $i = j$, $h_{ii} = \frac{|\nu_i|!}{2(|\nu_i|-2)!}$ and $h_r = \sum_{i=1}^{|L_r|} \sum_{j=1}^{|L_r|} h_{ij}$. For example, if robot r has observed 5 robots spread over two connected islands, $\nu_1^r = \{1,3\}, \nu_2^r = \{2,4,5\}$, and $\mathcal{E}_{1,2} = 1$ then $h_r = 10$. Note that since the localization, communication, and detection of other robots is deterministic no robot can observe more connections than actually exist, $0 \leq h \leq h_{true}$ (Assumption iv).

Lemma 2 (Robot Connection Convergence)**.** *Given two robots with sets of edges, $\mathcal{E}_r^-$ and $\mathcal{E}_s^-$ corresponding to connectivity measures h_r^- and h_s^-, Algorithm 4 will result in a $\mathcal{E}^+$ with measurement h^+ such that $h_r^+ \geq h^-$ and $h_s^+ \geq h^-$. Furthermore, at least one of the inequalities is strict if and only if $\mathcal{E}_r^- \neq \mathcal{E}_s^-$.*

Proof. When a robot merges data with another robot, the robot's connectivity measure h^- must either increase or stay the same. This is because the measure is a count of the total number of connection the robot is aware of, and the merged information must contain only new connections or redundant information since all robot localization is deterministic (Assumption iv). Since the merged set of edges, $\mathcal{E}^+$, now contains at least the same number of connections as each of the unmerged sets ($\mathcal{E}_r^-$ and $\mathcal{E}_s^-$), the resulting measure (h^+) must be greater than or equal to each of the measures of the unmerged sets h_r^- and h_s^-.

In order to reach a contradiction to prove the "only if" in the last statement of the lemma, suppose at least one of the inequalities is strict but that the two unmerged sets of edges are equal ($h_r^- < h^+$ or $h_s^- < h^+$, and $\mathcal{E}_r^- = \mathcal{E}_s^-$). The strict inequality implies that the connectivity measure increases, which can only happen if new connection information is contained in the second set of edges. However this contradicts the assertion that the two unmerged sets of edges are equal. Therefore a strict inequality is sufficient to conclude that the unmerged sets of edges are not equal.

Now we prove the "if" part of the last statement in the lemma. Suppose $\mathcal{E}_r^- \neq \mathcal{E}_s^-$ and neither inequality is strict ($h_r^- = h_r^+$ and $h_s^- = h_s^+$). Since the sets of edges are not equal, there must be information contained in one that is not in the other. By the properties of Algorithm 4 no edge information will be lost and $h_r^+ = h_s^+$. Therefore at least one h value must increase, which contradicts the assertion that neither inequality is strict. Therefore $\mathcal{E}_r^- \neq \mathcal{E}_s^-$ is sufficient to conclude that at least one inequality is strict. □

There are three atomic differences a robot can have between its observed graph and the true graph, which will be referred to as graph mistakes. The first mistake type is when a robot r does not know of the existence of another robot, robot s (i.e. s does not appear in L_r). The second mistake type is when a robot does not know that two robots inhabit the same island (this can refer to both itself and another robot or two other robots). This means L_r has two sets of robot ids that should be merged. The final mistake type is when a connection in the adjacency matrix $\mathcal{E}_r$ is missing, i.e. that the robot knows of two islands but is unaware the islands are within communication distance.

A result of Lemmas 1 and 2 is that when Algorithm 3 and 4 merge data with a robot with a mistake and another robot without that mistake, then a correction is guaranteed to occur. Changes in $d(L_r^-, L_{true})$ and h can be viewed as counts of these corrections.

Lemma 3 (Different Island Communication Frequency)**.** *Given two inhabited connected islands, E_1 and E_2, with all robots running Algorithm 1, every robot on E_1 will communicate infinitely often with every robot on E_2.*

Proof. The frequencies at which the robots reach a given point on the perimeter of the island is different for robots on different islands since their velocities are the same and the island perimeter lengths are different (see Assumption ii). This results in the fact that for robots on different islands every combination of perimeter positions will be realized infinitely often, including the locations where the two robots are within communication range. These locations must exist by the definition of the islands being connected. □

Lemma 4 (1-Hop Graph Corrections)**.** *Given two inhabited connected islands, E_1 and E_2, with all robots running Algorithm 1, every robot on these islands will update their observed graphs (G_r) to contain existence (L_r) and connection ($\mathcal{E}_r$) information regarding every robot on these same islands (E_1 and E_2) in finite time.*

Proof. At the beginning of the algorithm execution, each robot r on island E_1 only knows about itself. Running Algorithm 1, robots on island E_1 may discover other robots on its own island and run Algorithm 2, but will eventually finish and begin circling the perimeter. Initially, robot r is unaware of the following information:

Missing Information.

i. The robots on island E_2.
ii. The robots on its own island, E_1.
iii. The connection between robots on its own island, E_1.
iv. The connection between robots on island E_2.

Lemma 3 says that robot r will communicate with all the robots on island E_2 infinitely often. This make robot r aware of all the robots on island E_2 and therefore will correct the mistakes related to Missing Information i. Since Lemma 3 applies to all robots on E_1, every robot on E_2 will be aware of every robot on E_1. Therefore all robots on E_1 will be made aware of all robots on their on island, correcting any mistakes of category Missing Information ii

When robot r communicates with robot s, part of what is sent is the list of other robots sensed by robot s as well as their locations (see Communication Bundle in Section II). Using Algorithm 2, robot r will be able to determine if these other robots are also on E_1 or are on a different island. This, combined with the guarantee robot r will learn about all robots on E_1 results in all mistakes of type Missing Information iii being fixed. By reciprocity between E_1 and E_2, all mistakes of type Missing Information iv will also be corrected. □

Note that the correcting of the mistakes listed in Missing Information can be interrupted by communication with robots on a third connected island, E_3. However Algorithm 2 is guaranteed to complete in finite time, and there are a finite number of islands and robots, so there is a finite number of possible interruptions and all of them will last for finite times.

Theorem 1 (Algorithm 1 Completeness)**.** *Given a set of inhabited connected islands, E, where all robots are running Algorithm 1, every robot will discover the true graph structure (G_{true}) and list of robots on each island (L_{true}) in finite time.*

Proof. As a result of Lemma 1, as robots merge data the distance between the observed set of island names and true set of island names is monotonically non-increasing ($d(L_r^{t+1}, L_{true}) \leq d(L_r^t, L_{true})\ \forall r$). Similarly, the result of Lemma 2 says that merging data results in a connectivity measure that does not decrease ($h_r^{t+1} \geq h_r^t\ \forall r$). For both, if there is previously unknown information introduced by the merge the inequalities will be strict. Furthermore both measures are bounded, $d(L_r, L_{true}) \geq 0$ and $0 \leq h_r \leq h_{true}$. Since the difference between the true set of names and the observed set of names is both non-increasing and lower-bounded, it must reach a limit. Similarly the connectivity measure is non-decreasing and upper bounding and therefore will also reach a limit.

We first prove that all robots eventually reach a common Graph, and then show that graph must be equal to the true environment graph. In order to reach a contradiction, suppose that all robots graphs have reached their limit and that $G_r \neq G_s$. For this to be true there must be a robot s that has a mistake and a robot r that has knowledge that would correct this mistake (or vice versa). By Lemma 4 we know that by a finite time all robots will have corrected their graphs to include knowledge of their 1-hop neighbors. Furthermore, since there is a finite number of robots, a finite number of islands, and one-hop neighbors are periodically communicating, we know that in finite time a correction to a mistake that is known by robot r must eventually reach robot s. This is a contradiction and shows that $\exists T\ s.t.\ G_r^t = G_s^t\ \forall t \geq T$ and $\forall r, s$, so all robots eventually obtain a common graph in finite time.

Finally, in order to reach a contradiction, assume after all graphs have reached their limits $G_r \neq G_{true}$ for at least one robot. Since all limits must be the same, this common graph must contain one of the mistake types listed in Missing Information (see Lemma 4). In particular, there must exist robot r on island E_i with mistakes regarding a neighboring robot on island E_j. However, this violates the results of Lemma 4 and a contradiction is reached implying that the common graph must be the true graph. Therefore $\exists T\ s.t.\ G_r^t = G_{true}\ \forall t \geq T$ and $\forall r$. □

An analysis of the processing time required for all robots to discover the true graph structure (i.e. achieve the results guaranteed by Theorem 1), shows that the worst-case processing time grows as $O(n^4)$ (see Algorithm 1). Also, simulations indicate that the number of steps required per robot to discover G_{true} and L_{true} increases linearly O(n) with the number of robots in Figures 7 and 8.

IV. SIMULATION AND RESULTS

A. Simulation

Several simulations are run to demonstrate the efficiency and scalability of the algorithm. The first simulation configuration has 6 islands spread over a 40×40 grid and all robots have a communication range of 7. This is run with varying number of robots, spanning from 5 to 100. A second simulation configuration is also run with 8 islands spread over a 40×40 grid where all robots have a communication range of 5. This also is run for a range of number of robots spanning from 5 to 100.

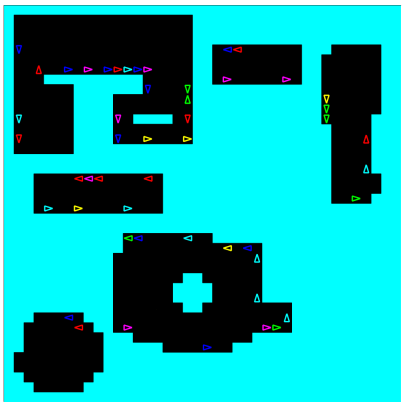


Fig. 5: Screenshot of simulation upon completion. Black areas are islands, blue area is water, and colored triangles represent robot positions and orientations. Six island environment with 60 robots.

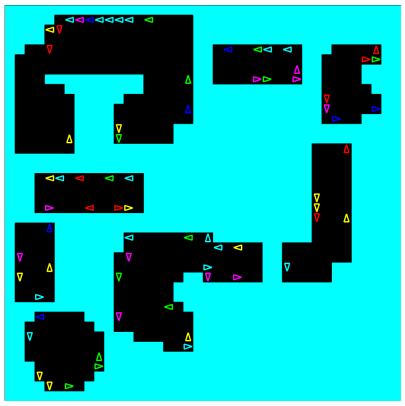


Fig. 6: Screenshot of simulation upon completion. Black areas are islands, blue area is water, and colored triangles represent robot positions and orientations. Eight island environment with 100 robots.

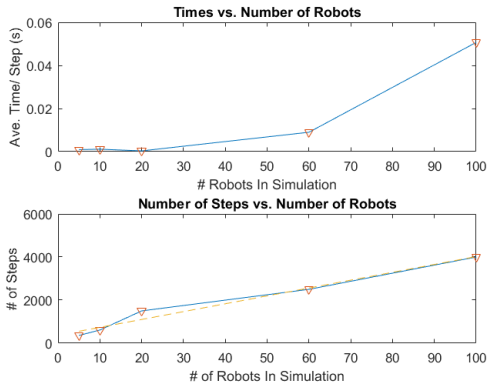


Fig. 7: 6 Island Simulation Results for Multiple Numbers of Robots (40×40 environment discretization with $D_{com} = 7$).

B. Results

Figures 7 and 8 show the results of simulating the two island environments shown in Figures 5 and 6 for a range of number of robots. The polynomial trend in processing time per robot is indicated, as is the linear trend for number of steps. These simulations were run on a ASUS N550JK Laptop with a Intel Core i7-4710HQ CPU @ 2.50GHz and 16GB of RAM.

V. Conclusion

In this paper, we propose a multi-robot mapping approach for many simple robots to discover the graphical structure of an environment. The robots have a limited communication range, GPS sensor, compass, and can only detect obstacles they are adjacent to. Using these restricted capabilities, we then demonstrate the completeness of the algorithm, and its guarantee that all robots' local estimate of the graph will converge to the true graph. Finally, we show the effectiveness of the approach with two simulation configurations run with multiple numbers of robots. An interesting direction for future work is to improve the efficiency of the algorithm,

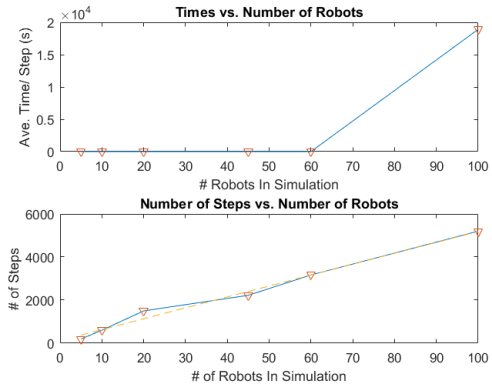


Fig. 8: 8 Island Simulation Results for Multiple Numbers of Robots (40×40 environment discretization with $D_{com} = 5$.

specifically focusing on the merging of the adjacency matrices. Additionally, the algorithm can be adapted to account for stochasticity in the robot's sensing capabilities. Furthermore, we hope to implement the algorithm on hardware to evaluate its real-world performance.

References

[1] L. H. Erickson, J. Yu, Y. Huang, and S. M. LaValle, "Counting moving bodies using sparse sensor beams," in *Algorithmic Foundations of Robotics X*. Springer Berlin Heidelberg, 2013, pp. 427–442.

[2] B. Tovar, F. Cohen, L. Bobadilla, J. Czarnowski, and S. M. Lavalle, "Combinatorial filters: Sensor beams, obstacles, and possible paths," *ACM Trans. on Sensor Networks (TOSN)*, vol. 10, no. 3, p. 47, 2014.

[3] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, "Map merging for distributed robot navigation." in *Proc. IEEE Intl. Conference on Intelligent Robots and Systems*, vol. 1, 2003, pp. 212–217.

[4] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[5] 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*, 2015.

[6] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1-4, pp. 403–430, 1987.

[7] F. Bullo and S. L. Smith, *Lectures on Robotic Planning and Kinematics*, 2015.

[8] F. Bullo, J. Cortes, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009.

[9] N. A. Lynch, *Distributed algorithms*. Morgan Kaufmann, 1996.

[10] J. W. Durham, A. Franchi, and F. Bullo, "Distributed pursuit-evasion without mapping or global localization via local frontiers," *Autonomous Robots*, vol. 32, no. 1, pp. 81–95, 2012.

[11] S. Thrun and Y. Liu, "Multi-robot slam with sparse extended information filters," in *Robotics Research. The Eleventh Intern. Symposium.*, vol. 15. Springer Berlin Heidelberg, 2005, pp. 254–266.

[12] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 945–955, 2006.

[13] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA Transactions*, vol. 46, no. 1, pp. 3–13, 2007.

[14] G. K. Fricke, G. Zhang, A. Caccavale, W. Li, and D. P. Garg, "An intelligent sensing network of distributed swarming agents for perimeter detection and surveillance," in *ASME 2010 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2010, pp. 741–748.

[15] A. Franchi, G. Oriolo, and P. Stegagno, "Mutual localization in a multi-robot system with anonymous relative position measures," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3974–3980.