EdinBots Team Description Paper for Rescue Simulation League 2019

Calum Imrie, Helmi Fraser, Hugo Sardinha,
Siobhan Duncan, Vibhav Bharti and Yaniel Carreno

Abstract

This paper presents the strategy devised by the EdinBots team to tackle the Rescue Simulation League - Virtual Robot Competition. Our approach is a biologically inspired hybrid strategy consisting of two levels: a stigmergy-based, intra-swarm data sharing layer used to guide overall exploration, which is fed by a point generating frontier-based exploration algorithm based on local sensing. For victim detection, agents will employ a dedicated classifier. For mapping, a global map will be constructed by merging together agent generated maps. Finally, a human operator can monitor the swarm and override control via a dashboard.

Keywords: *Swarm intelligence, stigmergy, pheromone, search and rescue, coordinated SLAM, classification, ROS*

1 Introduction

Swarm robotics is an increasingly studied topic, particularly in the context of search and rescue (SAR) scenarios. In recent decades, the autonomy and decision-making capabilities of multi-robot systems have improved dramatically. As a result of this, a paradigm shift is occurring within rescue robotics with a reduction in the number of humans in the loop. Operators are now increasingly free to perform higher level disaster assessment, as opposed to manual teleoperation. Owing to the highly distributed nature of the SAR domain, swarm techniques are a natural fit [1]. Applying such methodologies presents several unique advantages: system robustness to individual agent failure; scalability to a large number of agents; self-organization and super-linearity i.e. the performance of the whole is greater than the sum of its parts. Leveraging these advantages in conjunction with effective mapping and victim detection, we believe we can create a highly effective and robust SAR system [1].

2 System Overview

This section will outline the general principles and components of the system. First, a brief summary of how the Robot Operating System (ROS) will be used to control multiple robots. Secondly, an outline of the split nature of the system architecture.

[1] A detailed contribution of every team member to the proposed strategy, can be found at `https://sd2468.wixsite.com/edinbots/home`

2.1 Robot Operating System

ROS is rapidly becoming the *de facto* software platform for robotic applications. [2] Due to the decreased complexity of a simulation, we have chosen a more centralized approach for our multi-robot system. Our solution addresses multi-robot systems by using different *namespaces* to establish individual robot controllers, which are running on the same underlying *rosmaster* process.

2.2 System Architecture

The system consists of two main components: a global component *Ground Station*, and N instances of a local component for each *Robot*. The *Robot* component will generate potential way-points of interest, create a local map and detect victims on the fly. The *Ground Station* will be responsible for gathering data from each robot such as pose, maps and points of interest. Based on this, it will share relevant data between robots, allowing them to make more informed decisions. It will also be responsible for merging the local maps. The ground station will also allow a manual control override of any connected. Fig 1 illustrates these concepts, for one robot.

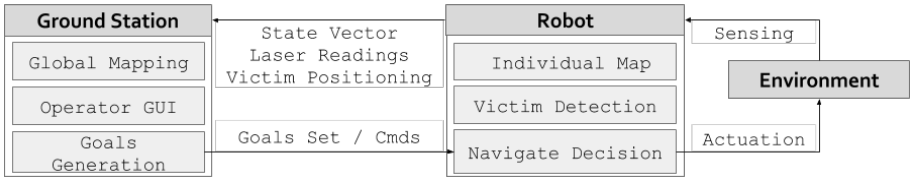

Figure 1: System Architecture. The main components of the system are the Ground Station and the Robot, which interact directly with the Environment. The Ground Station controls the robot set, optimising the robot navigation and victim detection

3 Multi-Robot Exploration Strategy

Exploration strategies for mobile robotics is considered an essential research problem for the implementation of complex mission in unknown environments. These strategies provide the robotic system with the functionality to explore unknown areas while creating a map of the environment and execute multiple actions of varying complexity. One of the key components of our system is the implementation frontier-based exploration scheme which allows robots to autonomously explore a complex environment.
Stigmergy is a phenomenon observed in nature where agents in a swarm can coordinate with each other by leaving clues in the environment, such as pheromone trails in ant colonies to coordinate the shortest path between a source of food and the next. A formal definition of stigmergy is as follows:

"Stigmergy is an indirect, mediated mechanism of coordination between actions, in which the trace of an action left on a medium stimulates the performance of a subsequent action" – Francis Heylighen [3].

In short, stigmergy can be broken down into the feedback loop displayed in Figure 2. An agent carries out an action which produces a mark within a medium, such as the environment or a virtual map, and this trace has a probability to stimulate an action in the same or an other agent which senses this trace. We plan on using a stigmergy inspired coordination strategy which is further explained in section 3.2.

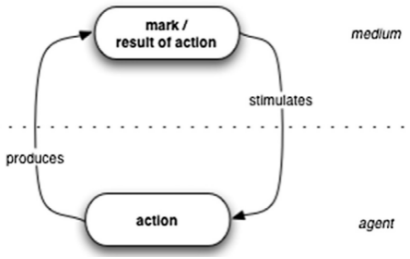

Figure 2: A visual description of the Stigmergy feedback loop between an agent and it's environment [3].

3.1 Frontier-Based Exploration

Frontiers are regions on the boundary between unexplored and explored space. Therefore, to acquire useful information from the environment, the robots have to navigate towards unexplored areas repeatedly during the mission. By repeating this process, the system maps the local area and expands this boundary between the known and unknown environment. There are multiple techniques for mobile robotics [4] based on frontier exploration. The main elements that we consider for the frontier exploration strategy are:

- `unknown-region:` area not covered by the robot's sensors
- `known-region:` area covered by the robot's sensors
- `open-space:` known-region which does not contain an obstacle.
- `occupied-space:` known-region which contains an obstacle.
- `occupancy-grid:` grid representation of the environment, cells contain a probability representing if it is occupied.
- `frontier:` boundary between known-regions and unknown-regions - a set of unknown points that each have at least one open-space neighbor.

We implement a Frontier-Based Exploration Strategy (FBES) using the robot's sensor system. Figure 3 (left) shows the general representation of the FBES, where each robot generates its own (i) map, (ii) frontiers, and (iii) set of exploration points on the boundaries with the unknown-region. The individual maps are lately merged to obtain the global map of the environment. The set of exploration points are inputs to a Goal Allocator (GA) strategy as well as the Pheromone techniques (see Subsection 3.2). The GA combines clustering strategies with the Pheromone gradients to define a "global" coordinate point

for each robot which guide the mapping. This avoids the risks of collisions and allows the optimal resource distribution (number of robots) to implement the exploration in the shorter time period. The coordinates are sent to the the robot move_base to execute the action. Figure 3 (right) shows preliminary results of the local FBES using three robots set in RViz. Each robot generates certain points (different colors) and those are the inputs to the GA, which calculate the point coordinates to be explored for the individual vehicles avoiding overlapping.

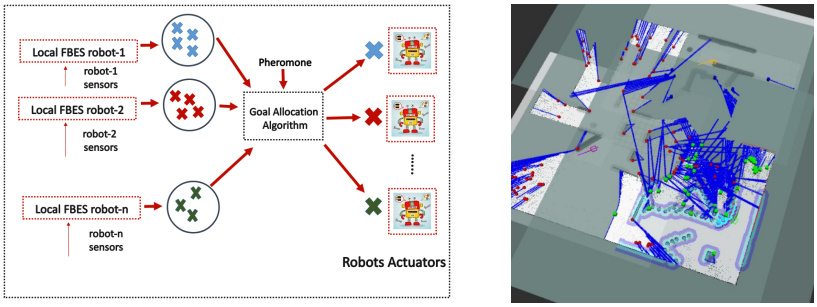

Figure 3: General goal allocation (right) based on Frontier-Based Exploration Strategy (FBES). The FBES is combine Pheromone techniques to implement the navigation approach. A depiction of the simulation environment (left) with multiple robots implementing the boundary strategy.

3.2 Stigmergy-Based Coordination

We plan to simulate a cloud system, implemented as a node on the *Ground Station*, which will serve a a virtual shared environment within which the robots can share and swap way-points generated by the frontier exploration outlined in Section 3.1 above. The goal is to allow the robots to coordinate themselves by sharing and self assign themselves to points of interest, without any need for a global planner.
The robots generate the points of interest which are then shared with the cloud system, which in turn returns a complete list of points in the local area of interest to the agent generated by all robots, rather than just its own points. These points are tagged by the ID of which robot has generated them, whether a robot is currently navigating to it and if so which one. This allows the robots to understand if there is another agent in their local area and areas of the shared maps they are exploring. The robot can then make a more informed decision of where to go next based on richer local information. In a more realistic situation, where there might be poor internet connection to the cloud service, the robots are able to fall back on the the individual frontier search algorithm as this just build on that.

4 Mapping

4.1 Motivation

The primary focus of this work is to find victims in a search and rescue environment. A key component to successfully rescuing all those trapped is to create the map of the environment. Beyond the scope of Robocup, the map is useful

not just for the robotic system but essential for the human users as well. Given the current state of robotics they should be expected as a significant assistive tool rather than a pure robotic solution (e.g. victims requiring medical attention should be attended to by a human).
This applies to indoor environments as well, even though an initial blueprint/layout is already available. Search and rescue environments can include natural disasters which will greatly alter the environment through structural damage.

4.2 Single Agent

The situation will assume that the robotic system does not have any information about the environment, including its initial pose. Furthermore the robots do not know where the other robots are. For now let us consider on how an individual robot in the system should approach this.
Simultaneous Localisation And Mapping (SLAM) [5] is a technique for the robot to not only create a map through its exploration but also know where it is with respect to its own generated map. It is a very difficult area and given that we have two different robots in our system we should identify the commonalities to be able to construct maps that can be brought together.
The robotic system consists of Hector drones and Pioneer ground robots. They are both equipped with a laser scanner, specifically Hokyuo. This therefore makes the ROS package **GMapping** [6] an appropriate choice. **GMapping** takes laser scan data and plots a 2D occupancy grid, and with each new reading builds upon this map. **GMapping** then, using odometry, determines where the robot is through particle filtering. Fig. 4 shows an example occupancy grid map output.

Figure 4: A sample output of **GMapping** [7]. The algorithm takes the data from the laser scan and build the map considering the open-space and the occupied-space.

4.3 Multiple Agents

The robotic system now have multiple robots producing 2D occupancy grid maps using **GMapping**, regardless if they are a Hector or Pioneer. Now these maps have to be combined to form an overall picture of the environment. Jiřì Hörner created an algorithm which when given 2D occupancy grid maps will stitch them together [8]. This has been implemented and is available as a ROS package, **multirobot_map_merge**.
The core idea is that the algorithm identifies key features from the individual

grid maps and then using feature matching finds shared points. The algorithm also contains a confidence interval to determine if the maps should be merged. Fig. 5 contains an example of how this works.

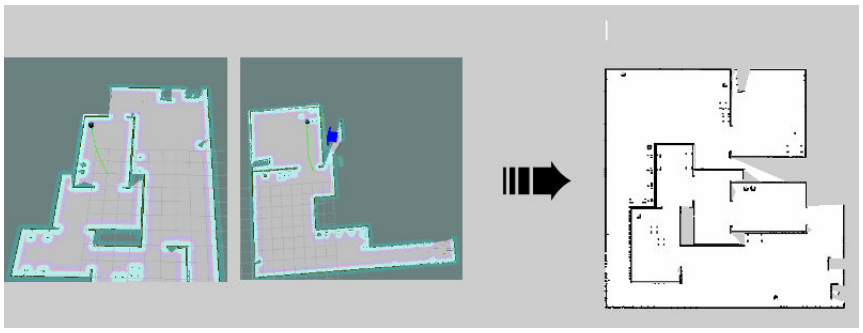

Figure 5: A sample output of **multirobot_map_merge** taken from [9]. Individual robot generates its own map and the map_merge produces the global map.

5 Victim Detection

Victim detection will be actively performed by the robotic agents. This will be primarily achieved via visual identification, though other modalities such as heat and sound will also play a role. The output of victim detection will be the position of the victim on the map, which can be overlaid on the map being constructed during exploration.
Ideally, the victims need to be classified into either alive or dead. This can be done by identifying some key, mutually exclusive features distinguishing the two, such as temperature. Similarly, if a victim is moving, it can be assumed that the victim is alive. Hence, the classification problem can be solved. Table 1 outlines some of these characteristics.

Victim	thermal camera	visual camera	Sound
Alive	high heat	hand wave	'help me' shouts
Dead	cold	motionless	no sounds

Table 1: Victim classification features and employed sensors. The robot's sensor system provide robustness to the victim state classification considering three types of sensors.

6 User Interface and Verification

6.1 Operator User Interface

Operator situational awareness is of paramount importance during search and rescue operations. It is therefore important to base design decisions around ease of use and efficient utilisation of space. Baker et. al [10] devised guidelines for efficient user interface design. These principles will be an integral part of our

design process. We would also like the GUI to be as lightweight as possible, so as to not influence the execution speed of the simulation. To this effect, we will reuse as much proven, "off-the-shelf", GUI components as possible.
Our operator *dashboard* will consist of a panel of labelled camera feeds from every robot in the environment, with the ability to quickly select one and teleoperate without a terminal. In addition to this, *rviz* will be shown alongside and this provides an overall view of the operational theatre. Finally, a multiplexed terminal will be available for anything requiring more complex commands.

6.2 Software Testing

Automated testing is a key component of making sure you have a reliable system when working with many developers. Unit tests help us avoid introducing errors into our packages and integration tests aid us to make sure different packages are interacting in correctly. We are using the ROS recommended testing frameworks: *rostest* [11] with *unittest* [12] and *gtest* [13] for python and C++ respectively. All development work is expected to have passed automated tests before it is submitted for code review. In addition to this we are using a git hub work-flow, where our workspace is divided into smaller repositories, allowing for work on different areas of the system to develop independently and where only the most stable version is used in the main build's workspace.
The main workspace and each sub-module have a master, integration and dev branch which contain code at different states of system reliability. The dev branch is used for the main work-flow, where new features, or fixes, are branched from here and merged back in when they have successfully passed automated tests and a code review from a team-mate. Toward the end of a milestone, and when a package has reach an acceptable standard, this dev branch is then merged with the integration branch, which can then be tested with the integration branches of the other packages to look for issues. When there are no issues, the integration branch is then merged with master which is considered to be a stable build of the system.

7 Conclusion

search and rescue is a clear trend in scientific research, particularly where the use of swarm systems is concerned. Our system proposes a hybrid approach based on virtual pheromones and frontier exploration to maximize the total explored area of an unknown environment, employing a heterogeneous swarm. Moreover, by introducing in-loop victim detection and map merging capabilities our system intends to provide a comprehensive pipeline for search and rescue missions.

References

[1] Micael S Couceiro, Patricia A Vargas, Rui P Rocha, and Nuno MF Ferreira. Benchmark of swarm robotics distributed techniques in a search task. *Robotics and Autonomous Systems*, 62(2):200–213, 2014.

[2] Micael S Couceiro, Rui P Rocha, and Nuno MF Ferreira. Ensuring ad hoc connectivity in distributed search with robotic darwinian particle swarms.

In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 284–289. IEEE, 2011.

[3] Francis Heylighen. Stigmergy as a universal coordination mechanism i: Definition and components. *Cognitive Systems Research*, 38:4–13, 2016.

[4] V Dayanand, Rahul Sharma, and Gireesh Kumar. Comparative study of algorithms for frontier based area exploration and slam for mobile robots. *International Journal of Computer Applications*, 77(8), 2013.

[5] Zhiwei Kong and Qiang Lu. A brief review of simultaneous localization and mapping. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 5517–5522. IEEE, 2017.

[6] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. GMapping by OpenSLAM. `https://openslam-org.github.io/gmapping.html`.

[7] Openslam:gmapping. `https://openslam-org.github.io/gmapping.html`. (Accessed on 02/26/2019).

[8] Jiřì Höner. Map-merging for multi-robot system. `https://is.cuni.cz/webapps/zzp/detail/174125/`, 2016.

[9] Ros: multirobot_map_merge. `http://wiki.ros.org/multirobot_map_merge`. (Accessed on 02/26/2019).

[10] Michael Baker, Robert Casey, Brenden Keyes, and Holly A Yanco. Improved interfaces for human-robot interaction in urban search and rescue. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 3, pages 2960–2965. IEEE, 2004.

[11] rostest - ros wiki. `http://wiki.ros.org/rostest`. (Accessed on 02/18/2019).

[12] unittest - ros wiki. `http://wiki.ros.org/unittest`. (Accessed on 02/18/2019).

[13] gtest - ros wiki. `http://wiki.ros.org/gtest`. (Accessed on 02/18/2019).