UNIVERSITÉ
LIBRE
DE BRUXELLES

Mémoire présenté en vue de l'obtention du diplôme de Master en Bioinformatique et Modélisation

Robot communication in swarm SLAM (Simultaneous Localization and Mapping)

Swarm SLAM, facing the challenge of sharing and exploiting map data in fully decentralized fashion

Loems Edwige

Master thesis submitted under the supervision of
Professeur Mauro Birattari

Academic year
2023-2024

In order to be awarded the Master's programme in
Bio-informatics and Modelling

Abstract

Swarm robotics is inspired by the behavior of social insects and animals that use local sensing and communication (directly or through their environment). There is no leader or external infrastructure, and they form a self-organizing system that allows them to perform tasks without having a global perception of the environment and objective. The main advantages of that system are that it is scalable, flexible, and fault-tolerant [15].

Following that concept, robots can be individually programmed to form a distributed and self-organized system. The challenge of swarm robotics resides in determining the individual behaviors resulting in cooperative behavior [6]. They are often based on the behavior of social insects, birds, or fish. For example, the Ant Colony Optimization (ACO) algorithm models the foraging behavior of ant colonies [16].

Because of its properties, a robot swarm is ideal for accomplishing missions in large unknown environments in which the risk that individual robots fail or are lost is high [35]. In particular, a robot swarm could autonomously perform simultaneous localization and mapping (SLAM). Most state-of-the-art methods for mapping often conflict with swarm robotics's characteristics (locality and absence of global knowledge) [15]. Some important questions must be addressed before effective swarm SLAM can be achieved: *"How should the swarm explore the environment and gather information? How should the robots share the information gathered? How should the information be retrieved and used to produce maps?"* [36]

The research conducted at IRIDIA in 2019 focuses on the first question and investigates the conjoint use of random walk exploration, the Gmapping algorithm, and the multirobot_map_merge algorithm to achieve swarm mapping [35]. Among the five variants of random walk evaluated, the ballistic motion provides the maps with the best quality due to the better ability of the swarm to cover the environment. Experiments were done both in simulation and with e-pucks robots. Mercator is an upgraded version of the Shero RVR, an education-tailed device with a large sensor set. One of the main differences between the RVR and the e-puck robot is the odometry capability. The robot can locate itself in space and whereas the e-puck used an estimation of its displacement [37]. This additional capability improves the mapping quality performed by a robot swarm.

The main focus is place on investigating the two remaining questions to achieve effective swarm SLAM. Most existing SLAM methods rely on external infrastructures to ensure inter-robot communication or localization [36]. It is, for example, the case in a 2010 study about the cooperative building of a chemical concentration map. The robots simultaneously send their sensor readings of the chemical concentration and position data to a remote computer [75]. One solution to ensure inter-robot communication in swarm SLAM is creating an Ad-Hoc network. It is particularly adequate for rough terrain that does not allow global communication. In most robot swarms, inter-robot coordination relies on uninterrupted access to situated, close-range communication of coordination messages between robots [73]. Using an ad-Hoc network for inter-robot communication simplifies the transition from experimental to realistic application scenarios. This has been used to simulate swarm exploration of extraterrestrial lava tubes [31]. Ad-Hoc networks are also used for search tasks of UAV (unmanned aerial vehicles) swarms [38]. UAVs can be equipped with hardware modules for radio communication between multiple UAVs. They are used for agricultural applications such as field coverage and weed mapping [2] [1]. Decentralized Wi-Fi communication between the Pi-puck can be achieved by establishing a Mobile Ad-Hoc Mesh Network between the robots [60]. The RVR is equipped with raspberry pi. The Wi-Fi modules allow the construction of an ad-Hoc network between the robots. Information gathered during the exploration can be shared through the network and exploited by other agents. The exploitation of the information by other agents provides two main advantages. The information can be exploited to coordinate their exploration strategies [19]. It also facilitates the production and recovery of the global map of the environment. It is optional to recover the whole swarm post-mission to collect the information collected by each agent. Each agent can exploit collected information about the environment and create a global map without external intervention. Swarm SLAM is a system with a high potential for mapping performance and efficiency and exhibiting scalability and fault tolerance characteristics. These features are essential for time-sensitive operations. Multiple studies investigate using a swarm of UAVs for collaborative and time-sensitive oil spill mapping [58] [4].

Contents

Abstract **iii**

I Introduction 1

II swarm SLAM 5

1 Background **7**

2 Approach challenges **9**
- 2.1 Multi-robot exploration . . . 10
- 2.2 2D occupancy grid . . . 10
  - 2.2.1 Gmapping . . . 10
  - 2.2.2 Map server . . . 11
- 2.3 Solving the map-merging and over-confidence problematic . . . 11
- 2.4 Communication . . . 13

3 System **15**
- 3.1 Simulated environment . . . 15
  - 3.1.1 Requirements . . . 15
  - 3.1.2 Ros node communication . . . 16
- 3.2 Real-world controlled environment . . . 18
  - 3.2.1 Robot . . . 18
  - 3.2.2 Camera . . . 19
  - 3.2.3 LiDAR . . . 19
  - 3.2.4 Ros node communication . . . 20

III Experimentation 23

4 Simulated testing **25**
- 4.1 Design . . . 25
- 4.2 Results . . . 27
  - 4.2.1 Simulation 1 . . . 28
  - 4.2.2 Simulation 2 . . . 30
  - 4.2.3 Simulation 3 . . . 31
  - 4.2.4 Simulation 4 . . . 32
  - 4.2.5 Simulation 5 . . . 34
  - 4.2.6 Simulation 6 . . . 35
- 4.3 Observation and interpretation . . . 36
  - 4.3.1 Simulation 1 and 2 . . . 36
  - 4.3.2 Simulation 3 . . . 37
  - 4.3.3 Simulation 4 and 5 . . . 37
  - 4.3.4 Simulation 6 . . . 38

5 Real-world experiment **41**
5.1 Design . . . 41
5.2 Results . . . 42
5.2.1 Experiment 1 . . . 42
5.2.2 Experiment 2 . . . 43
5.2.3 Experiment 3 . . . 44
5.2.4 Experiment 4 . . . 45
5.3 Observation and interpretation . . . 45
5.3.1 Experiment 1 . . . 45
5.3.2 Experiment 2 . . . 45
5.3.3 Experiment 3 . . . 46
5.3.4 Experiment 4 . . . 46

IV Discussion **47**

Conclusion **51**

Supplementary material **53**

List of Figures

3.1 rvr used during the experiments . . . 18
3.2 Module Step Down DC-DC . . . 19
3.3 Arena used for the experiments . . . 20

4.1 Simulation 1 and 2 arena . . . 25
4.2 Simulation 3 arena . . . 25
4.3 Simulation 4 arena . . . 25
4.4 Simulation 5 arena . . . 26
4.5 Simulation 6 arena . . . 26
4.6 Simulation 1; Seed 7; Legended maps . . . 28
4.7 Simulation 1; Seed 7; Individual maps merged . . . 28
4.8 Simulation 1; Seed 7; Maps . . . 29
4.9 Simulation 2; Seed 9; Maps . . . 30
4.10 Simulation 3; Seed 9; Maps . . . 31
4.11 Simulation 4; Seed 5; rvr2-rvr4 maps . . . 32
4.12 Simulation 4; Seed 1; Time frame 125 . . . 33
4.13 Simulation 4; Seed 3; Local map rvr3 . . . 33
4.14 Simulation 4; Seed 3; Global map rvr3 (left) and rvr1 (right) . . . 33
4.15 Simulation 5; Seed 1; Time frame 540 . . . 34
4.16 Simulation 5; Seed 8; Maps . . . 34
4.17 Simulation 6; Seed 7; Maps . . . 35

5.1 Display of the rvr inside the arena . . . 41
5.2 Experiment 1; Individual maps of (from left to right) rvr0, rvr2 and rvr3 . . . 42
5.3 Experiment 1; Global maps . . . 42
5.4 Experiment 2; Individual maps of (from left to right) rvr0, rvr2 and rvr3 . . . 43
5.5 Experiment 2; Global maps . . . 43
5.6 Experiment 3; Individual map of rvr2 (left) and rvr3 (right) . . . 44
5.7 Experiment 3; Map received by rvr1 . . . 44
5.8 Experiment 4; Successive frames of experiment 4 . . . 45
5.9 Experiment 4; Local map of rvr2 (left) and rvr3 (right) . . . 45

5.10 Simulation 1; Seed 7; Local maps . . . 54
5.11 Simulation 2; Seed 9; Local maps . . . 55

Part I

Introduction

Swarm robotics is inspired by the behavior of social insects and animals that use local sensing and communication (directly or through their environment). There is no leader or external infrastructure, and they form a self-organizing system that allows them to perform tasks without having a global perception of the environment and objective. The main advantages of that system are that it is scalable, flexible, and fault-tolerant [15]. This promising approach also face the challenge that is designing the individual behavior of the robots so that a desired collective behavior emerges [41].

There is no general methodology to predict the global behavior of a robot swarm based on the behavior of a single individual. With automatic off-line design, a control software specifically designed for a mission can be generated trough an optimization process. Research in swarm robotics has shown that automatic design is an effective approach to realize robot swarms [20] [52] [29] [44] [28] [9] [8] [71] [43] [22] [42] [7] [26] [49] [40] [33] [66].

Providing robots with the ability to accurately map an environment and simultaneously localize themselves within that environment can be achieved by solving the Simultaneous Localization and Mapping (SLAM) problem [65]. The main advantages of a swarm system make it valuable in the context mapping, specifically in unknown environments that evolve over time. The SLAM problem has mostly been explored in single and centralized multi-robot systems. In contrast to a swarm system, they cannot easily adapt to unexpected changes in the environment and are prone to failure in hostile environments [36]

Scaling a single robot SLAM to a multi-robot system pose the question of data sharing. Single-robot SLAM estimates are local in the individual robot reference frame. When multiple robots operate in GPS-denied environments, they need to solve the challenge of sharing situational awareness [46].

An approach to achieve distributed swarm SLAM is to reduce the amount of data to be shared. As highlighted by Kegeleirs et al. [23], promising candidates exploit the sufficiency of schematic map to map dynamical environments. In distributed mapping, the local maps can be shared within a cluster, allowing the robots to know their position in a shared partial map of the environment. This can be used to coordinate their exploration strategies to maximize the efficiency of exploration [16].

To take full benefits of multi-robot mapping, the data of different robots needs to be integrated to obtain global maps. When merging individually build local maps without any knowledge about their relative positions, maps are usually merged based on shared features to identify regions of overlaps. Techniques can be implemented to estimate the relative robots poses at the start or during the mapping process. The related locations can be used to combine maps based on occupancy grids [7]. This leave the question of how to treat conflicting data.

Part II

swarm SLAM

Chapter 1

Background

Providing robots with the ability to accurately map an environment and simultaneously localize themselves within that environment can be achieved by solving the Simultaneous Localization and Mapping (SLAM) problem [65]. There is two main approach to tackle the SLAM problem. In the Kalman filter based approach, the produced map presents the posterior probability of the location of identified features that are detected by the robot while exploring the environment. In the second approach, the most likely map is produced. The mapping task is solved using an algorithm insensitive to the data association problem based on the expectation maximization principle (EM) [10]. Solving the SLAM problem result in the production of high-quality maps of the robot surroundings [46].

Scaling to a multi-robot system pose the question of data sharing. Single-robot SLAM estimates are local in the individual robot reference frame. When multiple robots operate in GPS-denied environments, they need to solve the challenge of sharing situational awareness [46]. In multi-robot SLAM, there is a distinction to be made between raw and processed data sharing [64]. Both present their challenges, raw data sharing might scale up poorly as the huge amount of data could quickly become impossible to transfer, making it less suitable in swarm SLAM. As for processed data sharing, the common existing approach are centralized and rely on external infrastructure such as GPS or remote computers to assemble the different subsets of data [36].

A shared situational awareness can be achieved by using shared raw data to find the transformation between different robots. This requires deriving inter-robot constraints from perceptual information and optimizing the transformation from any robot frame to the base frame. This cooperative process is not resilient to infrequent overlapping between trajectories or the inconsistent moving directions in the overlapping area. The generation of reliable inter-robot constraints is founded on accurate pose estimation of neighbouring robots. However, by sharing accurate constraints, the system can correct localization errors and improve its overall SLAM performance [72].

Decentralizing the sharing of processed data can be achieved through the construction of a mobile ad-hoc network [14]. A Mobile Ad Hoc Network (MANET) can be defined as a collection of devices with wireless communications and networking capability that communicate with each other without the aid of any centralized administrator [59]. An ad-hoc network is considered in environment that may not allow global communication, causing the network topology to keep changing over time. Ad-hoc networking is a common solution to enable robots swarm communication [31]. The provided communication on peer-to-peer basis demands a routing protocol to send and receive packets and reveals the problem of communication in distributed multi-robot teams forming a mobile ad hoc network. There exist a wide range of solutions on how to route messages in ad hoc networks between multiple robots. The routing protocols can be divided into proactive (table-driven), reactive (on-demand) and hybrid categories. In

reactive protocols, the process of route discovery is only made on request. Reactive protocols requires less memory and relatively little control traffic overhead, making it preferable for multi-robot system. To improve the caused delay of packet transmission during the route discovery, an hybrid with proactive protocols that constantly maintain the relevance of the routes between sources and destinations as been proposed as an ideal solution for ad-hoc communication in multi-robot systems [69].
The traditional host-centric architecture is frequently unfavored over an information-centric or content-centric strategy. Opting for an information-centric solution set the focus on interested content data, rather than having to reference a specific, physical location where that data is to be retrieved from [54]. The superior architectural support provided by information-centric mobile ad hoc networks (ICMANET) explains its recent growth of interest. Information-centric reactive routing can use on-demand or controlled flooding to reduce the overhead caused by blind flooding. [53] These ability are assets to met to the scalability demand of swarm robotics. This was applied in field coverage and weed mapping by UAV swarms, using a Geo-aware broadcasting strategies with a maximum utility function. This protocol has been proven efficient in minimising the number of messages transmitted and maximising the utility, without affecting the coverage efficiency [1].

An approach to achieve distributed swarm SLAM is to reduce the amount of data to be shared. As highlighted by Kegeleirs et al. [36], promising candidates exploit the sufficiency of schematic map to map dynamical environments. In distributed mapping, the local maps can be shared within a cluster, allowing the robots to know their position in a shared partial map of the environment. This can be used to coordinate their exploration strategies to maximize the efficiency of exploration [19]. In graph-based SLAM, the local frames of data and the relative spatial relationships between local frames is maintained using measurement that depends only on the relative location of two state variables. A bundle adjustment using consensus optimization is run to keep a globally consistent estimate of all map data [45].

In swarm SLAM, this consensus is controlled by locality. Reaching a consensus in a decentralized system requires additional delays and data sharing [36]. A divide and conquer approach could solve this dynamic optimization problem [76] [36]. Majcherczyk et al. proposed a solution for the fusion of collective perception in resource limited mobile robot swarms where a semantic map is constructed by consolidating multiple observations of the same objects into single ones. The imperfection of the classifier and mislabeled objects are corrected through a voting mechanism. This mechanism result in semantic maps whose result is superior of the one constructed by a single robot [57].

The real-world implementation of such system might require a more sparse distribution of the swarm that would limit inter-robot communication [73]. Many industrial projects often rely on a centralized communication infrastructure to execute their solution with multiple robots, neglecting the principal idea of swarm robotics of distributed decision making [68]. It is the case in the cooperative chemical concentration map building presented by Turduev et al. where a remote computer is needed to combine the sensor readings of the robots and form a real-time map of the chemical gas concentration in the environment [75].

Chapter 2

Approach challenges

The proposed approach share similarities to a partial swarm SLAM technique, whereby sections of objects discovered by different members of the swarm are stitched together considering an error rate and broadcast to members of the swarm. However, in this approach, only a small number of leading agents contribute to the building of the maps, that is broadcast to follower agents for path planning [56].

Another existing approach revolve around a distributed algorithm for sharing and fusing occupancy grid maps among robots in such a way that each robot's map eventually converges to the same global map of the entire environment. Each robot's occupancy grid is updated based on occupancy probabilities computed from its own laser range sensor measurements and the occupancy grid map information broadcast by robots that are within a distance. This system's best performance require the agents to have a high rate of encounters with other robots and with the boundaries of the free space [62]. Building a single global map without requiring a central authority require concurrent access and modifications to the map data. This pose the problem of reaching a consensus regarding conflicting values [12].

The objective of the proposed approach is to achieve a distributed swarm SLAM through distributed mapping. The investigated distributed mapping approach revolves around the communication of the agent's global map to its neighbours using image messages. Each agent compute its own global map that is dynamically and locally updated using its own sensing information as well as the received image messages from other agents. This approach encounter two main challenges: solving a map merging problem to update the agent's global map and restraining the agent's over-confidence.

The map merging problem is the well researched problem of the creation of a consistent global map from local ones that are produced by different agents. The solution is apparent when choosing a static approach where a global map is created after each local agent explores its own environment. Yet, a efficient swarm SLAM system would require a dynamic approach, where global and local maps are produced simultaneously [18].

To take full benefits of multi-robot mapping, the data of different robots needs to be integrated to obtain global maps. When merging individually build local maps without any knowledge about their relative positions, maps are usually merged based on shared features to identify regions of overlaps. Techniques can be implemented to estimate the relative robots poses at the start or during the mapping process. The related locations can be used to combine maps based on occupancy grids [10]. This leave the question of how to treat conflicting data.
2D map matching is an algorithm based on area segmentation. The robots 2D occupancy grid maps are transferred to an area graph representation and a consensus is reached by voting in that space [30].

The multi-robot map merging problem is often associated with the assumption that the concerned maps are of similar quality. A Convolutional Neural Network (CNN) method for map fragment classification is proposed by Andersone. I. to allows the quality evaluation of occupancy grid maps without the need for ground truth maps. This method can be used for the overall map quality evaluation or for sub-regions [3].

2.1 Multi-robot exploration

Multi-robot exploration can be addressed as a decentralized task allocation to coordinate robots in the exploration mission while exchanging only their positions [5]. This model is put into practice with the mapping of weeds by UAVs recruiting each other towards areas of possible interest [1].

Kegeleirs et al. investigated the exploration capabilities of robot swarm into the context of mapping by comparing the quality of maps produced using five variants of random walk. The different mapping results were assessed by merging the agent's local maps into a unique global map in a static way, favoring the ballistic motion variant of the random walk. The experiments were conducted whether the initial position of the robots is known or not, making the reasonable assumption that the initial relative position of the robots could be known *a priori* when the robots are deployed in a fixed location [35]. Different exploration schemes were also explored in the context of automatic modular design methods [70]

To focus on developing a dynamic solution for the multi-robot map merge problem, the proposed model is constructed around a similar ballistic motion variant of the random walk, with the constant assumption that the initial relative position of the robots are known.

2.2 2D occupancy grid

As previously mentioned, there are many benefits to the use of processed data in the context of swarm SLAM. In the proposed method, map data is shared between agents through image messages in which the image messages are a result of a map-merging process. This model also takes advantage of the use of semantic maps to reduce the global amount of shared data. Occupancy-based maps are frequently preferred by researchers for their metric representation and ability to present information about unexplored regions [18]. Occupancy grids are particularly effective to map unstructured environments where features extraction is hard to perform [10].

The development of multi-robot SLAM systems for 3D LiDARs gains in popularity [17]. While these systems can consistently improve the accuracy of the mapping process, there use goes hand in hand with the use of more complex robots, making it less suitable for swarm SLAM.

2.2.1 Gmapping

The rospackage Gmapping is a laser-based SLAM algorith proposed by Grisettiet al [25]. The combination of optimized PF algorithm and improved resampling process results in a good map's quality [67]. Gmapping updates the pose on each processed particle based on the estimation of odometry and perform a laser scan match to correct the estimation of pose in the map of each particle [24].

In order to limit the LiDAR capacity, the maxUrange rosparam is set to 0.4m. This reduce the maximum usable range of the laser by cropping the beam to 0.4m. This limitation is necessary for the interpretation of experiments performed in smaller environment, as using the maximum range of the LiDAR would produce unrealistic results. In this proposed system, each agent initial position and orientation is know. These parameter are used to compute a corrected odometry, allowing the swarm to share a coordinate system and making all the produced maps easily comparable.

2.2.2 Map server

The ros map-server package provide a map-server ros node of type map-saver to save a map from SLAM mapping service to disk [23]. It offer a solution to retrieve map data from the gmapping SLAM algorithm and generate a pair of file.

- *YAML file*; Describe the map meta-data
- *Image PGM file*; Encode the occupancy data performing a trinary interpretation (interpret all values so that the output ends up being one of three values).

The image file encodes the occupancy state of each cell of a predefined size environment using an alternations of three colors; white for the free state, black for the occupied state and gray for the unknown.

In order to minimize the CPU load, the re-spawn delay is set to 2. This allows every agent to update its local map image every 2 seconds. Considering the global maximum speed of the swarm, a 2 seconds delay should not result in the loss of a significant amount of data.

2.3 Solving the map-merging and over-confidence problematic

The proposed solution revolve around the conjoint use of four maps by each agents.

1. Saved image by each agent, referred to as the Agent local map
2. Agent last accepted image message
3. Agent global received map; Contain information from the accepted image messages
4. Agent global map; Contain information from the agent's local map and the received map data

Each global map is dynamically constructed during the exploration through the same merging process. The *Agent global received map* is empty at init time. Upon accepting a first message, the map is updated and the *Agent global received map* is identical to the *Agent last accepted image message.* The merging process start at the acceptance of a second message. The *Agent global received map* is updated by merging the accepted image to itself. The *Agent global map* is identical to the *Agent local map* at init time. When the *Agent global received map* is generated, the *Agent global map* is updated by merging the *Agent global received map* and the *Agent local map.*

The merging function can be described as a succession of four steps and always involves two provided images.

1. The images are read using the python cv2 library.
2. The second step consist of reshaping the images without data loss to unsure that they exhibit coherent shapes. If needed, the image is restructured from a two-dimensional array to a three-dimensional array, where the first two dimensions represent the height and width of the image, and the third dimension represents the number of channels. The third dimension represent the unique channel characteristic of a gray-scale image. Adding a third dimension render the image more suitable for image processing algorithms.

3. Combining the images. The two images are blend with equal weights.
4. Application of a median filter using a disk-shaped structuring element with a radius of 1 using the python skimage library.

The chose size of the median filter is an arbitrary choice based on visual inspection. It was made on the hypothesis that it could reduce the noise, principally linked to presence of other agents in the system.

Artificial limitation are implemented to lean forward more realistic results. Each agent subscribe to the odometry of the others by removing itself from the possible list of id. Upon receiving a map message, it verify that the distance with the source is smaller than 1m. Otherwise, the message is not considered. It is to be expected that the development of the control software in simulation will suffer from a drop in performance when ported to physical robots. This reality gap is a critical issue in the manual and off-line automatic design of control software for robot swarms [50] [51]. This problem is reminiscent of the generalization problem faced in machine learning. Constraining the representational power of the control software is a key factor in crossing the reality gap [34].

Sharing the agent global map present benefits over sharing only the current sensed data or even its current local map [39]. This processes allows the broadcasting of information beyond the agent at its source. This come at the cost of risking an over-confidence of data.

A second condition is implemented to limit the over-confidence of data and the processing of low utility messages. The *Agent last accepted image message* is overwritten by each accepted image message, i.e. the ones that contribute to the *Agent global received map.* For a message to fulfill the acceptance conditions, it is compared to the *Agent last accepted image message.* The root mean square deviation between the images is computed using the *sewar python package.* A fixed threshold determine the acceptance of the received images. The root mean square deviation measure was chosen for the facility of its implementation. Other measure were tested (image hash algorithm, brisque image quality assessment) but did not suggest an adventage to the RMSD computation. A similar approach was used by Cunningham et al in which two distinct maps enables the robots to maintain a consistent SLAM solution with a local map. A second augmented local maps is used to summarized the received information. Maintaining a the local map enables the system to prevents double-counting [13].

In the third step of the merging function, the images are bind with equal weights. This merging process could allow a classification of the received occupancy data, without requiring additional space to keep all the received messages in memory. It is based on the suggestion that the successive merges could produce a map, where the gray-scale level of each pixels could be interpreted in the following way. The level will increase when converging information is received, and decrease otherwise. The level is lower is the information has not been received as frequently as other information. The level will also be lower is the information is older and has not been corroborated since.

This observation can be compared to the distinction made by Majcherczyk et al. between an observation and consolidation coverage. The observation coverage consider an object covered if at least one robot annotated the object while the consolidation coverage considers an object covered if there exists a consolidated annotation for the object in the shared memory [57]. The possible distinction made through the merging process between old and new information could contribute to diminish the over-confidence by providing the system with a mean to subtract out old information [13].

2.4 Communication

The collective behavior of a robot swarm emerges from the interaction that individual robots have with their peers and their environment [63]. This makes a necessity for the control software to provide a support for local communication [27]. The robots under consideration for the testing of the proposed system are not equipped with a communication module but are each equipped with a raspberry-pi. Two alternatives can be considered to simulate a communication system.

The first solution would be to take advantage of the WiFi modules of the robot's raspberry-pi to establish a mobile ad-hoc network. This procedure is well-documented [60] and has been tested with the considered robots. For an easier implementation of the system in real-world conditions, a second solution has been however preferred, in which the robots use a rostopic to exchange data.

Chapter 3

System

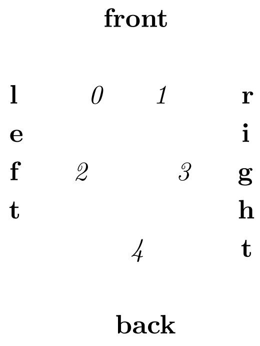
The proposed system is developped as a *Ros-based system.* A roscore (collection of nodes and programs) is running in order for the ros nodes to communicate.

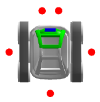
It has been developped base on Mercator, a proposed hardware and software architecture for experiments in swarm SLAM [37].

3.1 Simulated environment

3.1.1 Requirements

An ARGoS node centralizes the control software. The control software is base on the software architecture of Mercator, that required downgraded version of Argos (beta 48) and lua (lua 5.2) [37]. The control software was developped under the Ros noetic distro, that require ubuntu 20.04. The system also require the ros packages slam gmapping, explore-lite and map-server as well as the python library cv2, skimage and sewar.



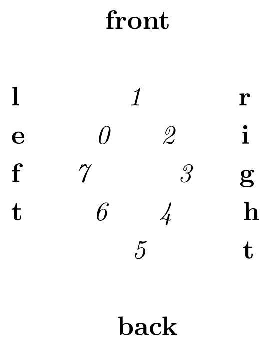


The robot developed in the software architecture (RVR) is equiped with 5 LEDs. The color can be set with the *LEDsActuator* listed in Table 3.1. The position of the LEDs is illustrated above. In order to simulate localization, a sixth virtual led is used to represent the robot's position and detected by the *omnidirectional camera.* By default, the modification of this sixth LED is not enable. The provided actuator needs to be modified in order to allow the detection of different LED color by the *omnidirectional camera sensor*

Sensor	**Actuator**
CCI_RVRProximitySensor	*CCI_RVRWheelsActuator*
CCI_RVRLocatorSensor	*CCI_RVRRGBLEDsActuator*
CCI_RVRQuaternionSensor	
CCI_RVRIMUSensor	
CCI_RVRLidarSensor	
CCI_RVRColoredBlobOmnidirectionalCameraSensor	

Table 3.1: Class of used sensor and actuator

The actual positioning of the proximity sensors on the real rvr differ from the last modification of *CCI RVRProximitySensor* provided by *argos3-rvr*. Because the proximity sensors were not used for the real-world implementation of the controller, this issue has not been addressed. The proximity sensors are still considered placed as bellow.



As mentioned previously, the assumption is made that the robot's initial position and orientation is known. For simplification purpose, the robot's used an identical initial orientation. The odometry however, as to be corrected according to the variation of initial positions in order to generate consistent maps. This correction is made by placing the robots at equal distance and correcting their computed odometry by a factor proportional to their id number.

As for the LiDAR, an artificial limitation is made for the usage of the proximity sensor, reducing its maximal range to 0.4m.

3.1.2 Ros node communication

The architecture of the Ros-system used for simulation and real-world experiments present a similar backbone. To avoid overwritting, each node is defined by an unique name or set to anonymous. In the following Table 3.2, the X represents unique nodes and topic's names, each robot replacing it by its unique id.

	Random Walk_X	***Slam_Gmapping_X***	***Map_saver_X***	***Images__X***
Flag_X	*Published*			*Subscribed*
Odom_X; TF	*Published*	*Subscribed*		*Subscribed*
Proximity_X	*Published*			
Scan_X	*Published*	*Subscribed*		
Common_map_publisher	*Published*			*Subscribed*
Map_global_X	*Subscribed*			*Published*
Occupancy_grid_X		*Published*	*Subscribed*	
Map_metadata_X		*Published*	*Subscribed*	
Entropy_X		*Published*		

Table 3.2: Nodes' published and subscribed rostopics; Simulation

The robots perform a random walk with obstacle avoidance in order to explore the environment. The obstacle avoidance rely on the proximity sensor outputs. The *omnidirectional camera sensor* returned the robot's perceived color in its close environment. The color detection is used to indicate the perception of another robot. A flag is published to indicate that the exploration has been initiated.

Each robot is responsible for the production of its local map. The *Slam Gmapping node* subscribe to the computed odometry and scan readings to publish map data as an occupancy grid, map metadata and entropy. This map data is saved locally as a pair of file with map saver.

At the beginning of the exploration, the local map saved as a pgm file is read by the *Image node* and publish it as image message with its id as header on *the Map global topic.* The *random Walk node* subscribe to the topic. Publish the receive message unto another topic, *common map publisher*, subscribed by all agents. This topic is publish only when perceive a red led is perceived, corresponding to the perception of another robot.

The published *flag topic* ensure that the images processing steps only begins when the scan topic is published. This avoid the unnecessary dying of the process because of a lack of access to map images.

The *Images node*, treat the received images message from the subscribed *Common map publisher topic.* It ignore its own message by checking the associated header. Upon receiving a message, as it is subscribed to every agent's odometry, can check the distance between its position and the source of the message. This ensure an artificial limitation, the messages are processed only when received within a reasonable distance.

Considering that all messages satisfy the distance requirement :

- If a first message is received, it is accept. The robot proceed to save the last image message received as a pgm file and merge the accepted message with its local map to send on the map global topic instead of the previously send local map.
- From the second message, it compare the RMSD between the first message received, the acceptance depending on a fixed threshold. If accepted, the last image received is saved as a pgm file. It proceed to merge the received image with the previous one and write it as well as merge the compilation of received image messages and the local map and publish it to map global.

- From the third message, it compare the RMSD with the pgm file of combined received maps. If accepted, it overwrite the last received file, and merge the received message with the previously combined received map. It then proceed to merge the new compilation an the local map to update the published map global message.

3.2 Real-world controlled environment

3.2.1 Robot

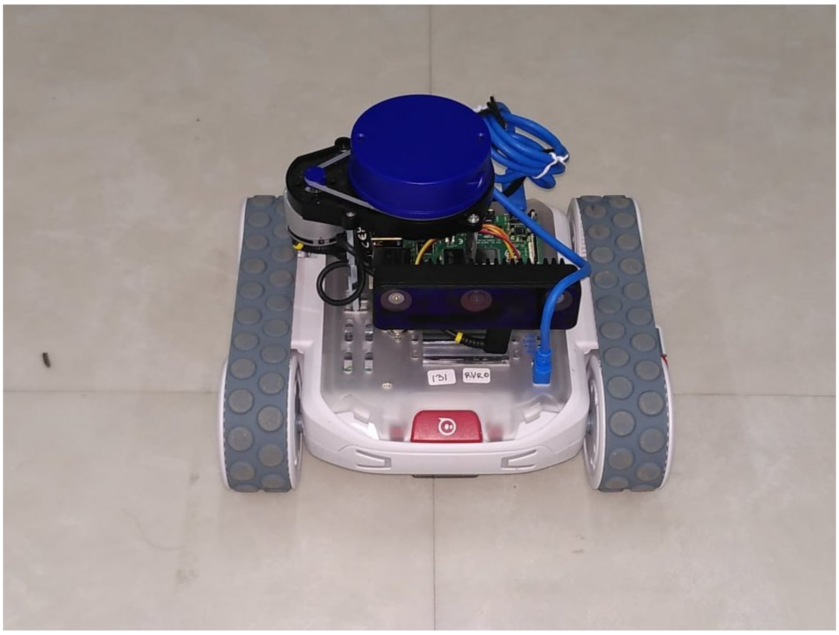


Figure 3.1: rvr used during the experiments

Four distinct robots were used for the real-world experiments. They have fixed id for the purpose of the communication protocols. Each rvr has a list of all existing id. This list can be exhaustive and the behavior of the controller is not impacted by the presence of the robot's id in the environment. If the id is not listed however, the message will fail to be considered by another robot.

The robot's hardware architecture is simple and slightly differ to the one described in Mercator [37] [32]. The robot is equipped with only three components: a raspberry-pi, a LiDAR and a camera. The base robot is the Sphero RVR all-terrain robot [74]. A more complex hardware architecture was initially considered, with the advantage of presenting additional features (two sets of camera, proximity sensor and additional an LEDs strip). These additional components require the power of a second raspberry-pi. A *Module step down DC-DC* is used to allow a spare battery to power it as illustrated in Figure 3.2.

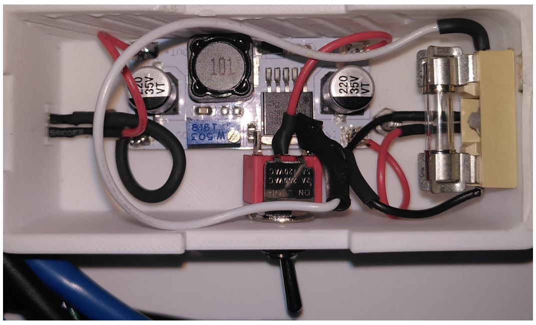


Figure 3.2: Module Step Down DC-DC

The chose to conserve the initial simpler hardware architecture was made to develop a system for which a bigger swarm float was available. The use of a simpler robot is also in accordance with the robot swarm principles.

It was observed that the *Rvr_async_driver_X* node, required to actuate the leds and wheels and obtain the odometry and Imu sensor data, dies more frequently when launched too prematurely from the *Random_Walk_X node*. This could be an issue related to the capability to prevent the rvr from entering a sleeping mode, thus leading to a *ROSInterruptException*. The driver has been modified to always display red LEDs and enabling its detection by other robots. This result in a loss of information regarding the robot's status that is communicated through the LEDs.

Some problems also seemed to occurred while using *Sphero Rvr* that were previously used for other experiments. The UART communication between the robot and the raspberry-pi seemed to be less consistent than when using new robots. This problematic seem to occur less often when the battery percentage is higher than 80%. These observation could pose the problematic of the general longevity of the *Sphero rvr robot*.

3.2.2 Camera

The camera used for the experiments is the *Luxonis, OAK-D camera*. This camera combine a stereo depth camera and high-resolution color camera. Currently, only the high-resolution color camera is used to detect other robots in the environment. This detection is made by analysing the camera output frames, cropping a window of observation and counting the number of red pixels. A threshold number of pixels is defined for the robot to consider having perceived another robot. This method could present the disadvantage of having to fine-tune the threshold to fit the environment.

3.2.3 LiDAR

IRIDIA's Robotics Arena [48] initially envisaged for the experiments had walls too low to be detected by the LiDAR. The problem was solve (as illustrated by Figure 3.3) by creation of detachable panels to the existing structure. The panels were created using spare material available and cut with a lasercutter to obtain desired sizes.

The LiDAR used for the experiments is the YD LiDAR X4 [77]. The exploitation of the LiDAR outputs face some challenges. When the frequency parameter is set to 10, 505 ranges are returned, corresponding approximately to 360 degree readings by 0.7 degree on the X4 LiDAR. The initial LiDAR positioning on the robot set the range[252] at the front, range[378] on the left, range[126] on the right and range[505] at the back. This observation is inconsistent with the information provided by the LiDAR data-sheet.

Using the default parameters and a frequency of 10, approximately 20% of zero-values were returned. Unfortunately, these ranges returning zero-values were clustered at the front of the robot, impacting its ability to detect obstacles in front of it. The percentage of non-zero values significantly increased when increasing the frequency to 12, giving the possibility of fine-tuning the random walk to this new set of ranges [21]. These zero-values are less problematic when using the simple version of the rvr, where the lidar is rotated 90 degree to the right with respect to its initial placement. This set the range[378] at the front, range[505] on the left, range[252] on the right and range[126] at the back. The chose was made to retain the frequency value to 10, integrating zero-values to the readings and providing noise to the experiment. Although these problematic does not appear to affect the mapping capability of the robot, they seem to interfere with its obstacle avoidance capability, making it prompter to fail its obstacle avoidance.

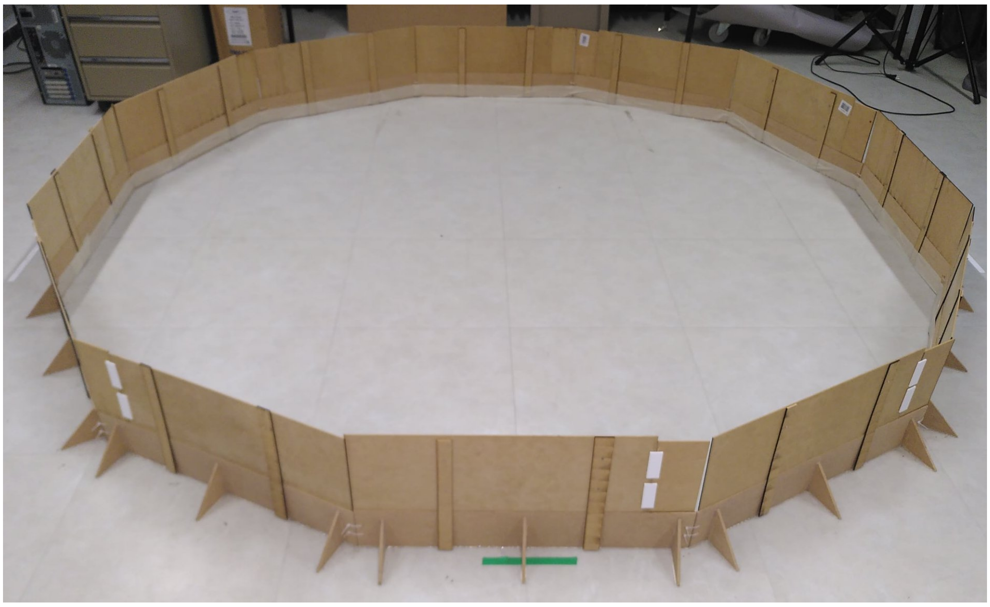


Figure 3.3: Arena used for the experiments

3.2.4 Ros node communication

	Random Walk_X	***Oak camera_publisher_X***	***Ydlidar lidar_publisher_X***	***Rvr_async_driver_X***
Flag_X	*Published*			
Odom_X; TF				*Published*
Scan_X	*Subscribed*		*Published*	
Common_map_publisher	*Published*			
Map_global_X	*Subscribed*			
Red_led_detected_X	*Subscribed*	*Published*		
Wheels_speed_X	*Published*			*Subscribed*
Rvr_rgb_led_X				*Subscribed*
Imu_X				*Published*

Table 3.3: Nodes' published and subscribed rostopics; Real-world controller; The nodes ***Map_saver_X***, ***Slam_Gmapping_X*** and ***Images_X*** are launched as well but absent from this figure because they exhibit the same architecture in both controller

The presence of multiple host require the configuration of a *Ros Multi Master*. This enable the connection between the robot's Ros environment. Each device is also synchronized to the main workstation and to ensure the cohesion between the robots environment.

The built *Oak camera publisher* rely on the DeptAI pipeline to obtain the camera output [55]. The frames are processed as previously described to indicate the perception of another robot in the environment through the publication of the *Red led detected topic.*

The Ros package *ydlidar_ros_driver* was used to publish the LiDAR ranges. This package depends on the *YDLidar library.* The *frame_id* parameter has been changed to the corresponding gmapping parameter. The robot_id has been added to both the *frame_id* parameter and rostopic name.

The Sphero SDK for Raspberry Pi ,*Rvr Async Driver* manage the rvr actuator and sensors. This require to enable the raspberry-pi UART communication. The rvr_rgb_led_X callback is used to set the led color in red in all situations. The mapping process require the addition of TF broadcasting using the *ros package tf* to publish the relative pose and coordinate to the system. This assure relationship between coordinate frames overtime.

Part III

Experimentation

Chapter 4

Simulated testing

4.1 Design

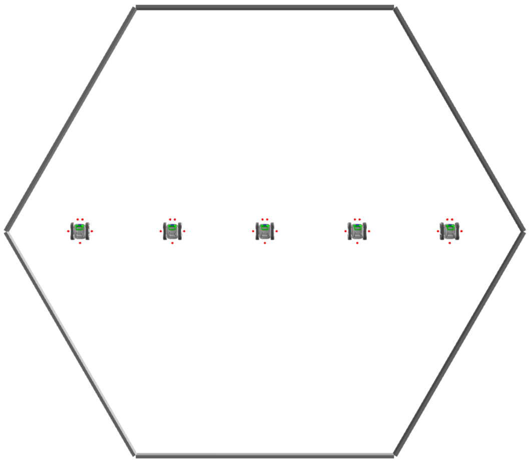


Figure 4.1: Simulation 1 and 2 arena

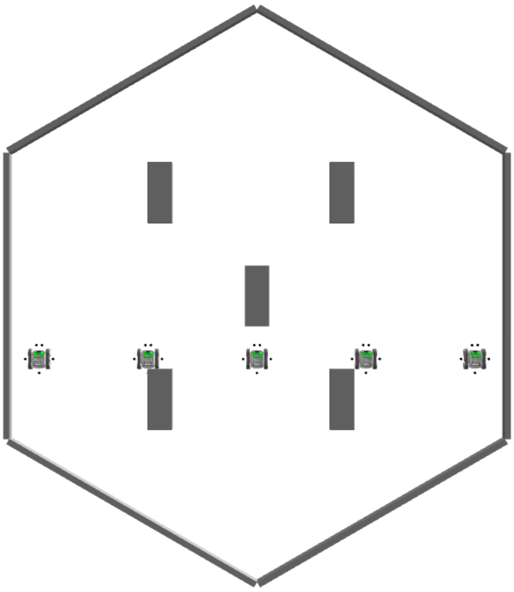


Figure 4.2: Simulation 3 arena

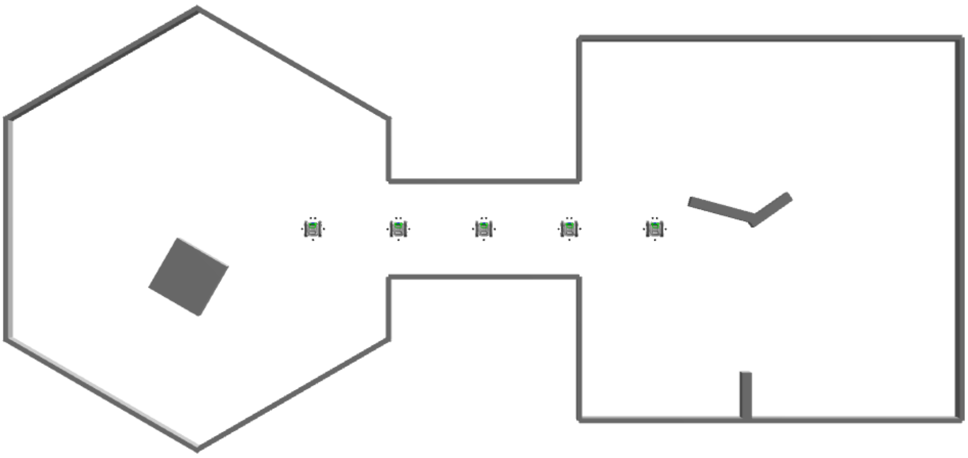


Figure 4.3: Simulation 4 arena

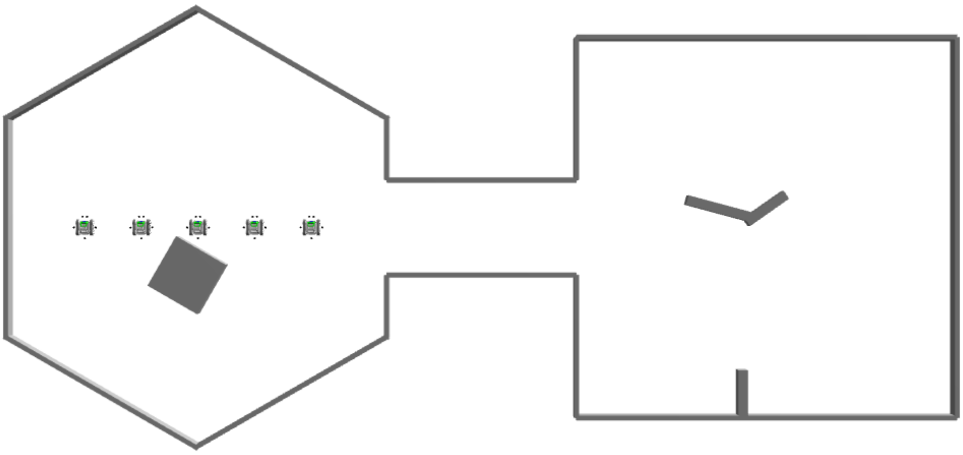


Figure 4.4: Simulation 5 arena

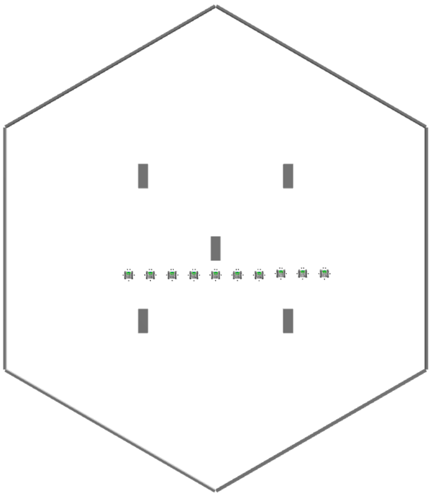


Figure 4.5: Simulation 6 arena

Six distinct experiments were generated to create a variety of testing environment. To avoid the confusion with the real-world experiment, these will be refereed as simulation 1-6. Each simulation is repeated 10 times with seed ranging from 1 to 10. The arena were designed using the graphical Argos arena editor tool. The orientation of each rvr is identical in all performed simulations for to allow the cohesion of the generated local maps.

The x and y axis of the coordinate system of the Argos simulator and the ros gmapping package are inverted. The Figure 4.1 to 4.5 are under the Argos coordinate system. The figures 4.6 to 4.17 and 5.2 to 5.12 are under the gmapping package coordinate system with the exception of figure 4.15 and 4.12 (frames capture from the Argos simulator, obey to the Argos coordinate system).

The simulation 1 and 2 only differ by the value of the rmse threshold (respectively fixed at 8 and 4). The arena used in both simulation is represented in the Figure 4.1. The radius of the hexagon is approximately 2.5 meter. The experiment duration is 2000 times steps. These simulations integrate 5 rvr with an equal spacing of 0.9m.

The simulation 3 is performed under similar circumstance than the first two. The arena is similar and has the same size but is permuted by 180 degree. 5 rectangular obstacles were added and the initial x position of the rvr are decrease by approximately 50cm to allow the position of the center obstacle. The rmse threshold is set to 8.

The simulation 4 and 5 both take place inside the same arena composed of two rooms. The experiment duration is increased to 5000 time steps to accommodate the increased size of the arena. The size of each room is similar to the size of the previous arena. The rmse threshold is remaining at 8 but the 5 rvr position differ in both simulation. In simulation 4, the 5 rvr initial position are identical to the initial position used to perform the first two simulations. In simulation 5, the 5 rvr are translated 5m alongside the y axis and the spacing between them is reduced to 0.6m to ensure the initial deployment in the first room.

In the sixth simulation, the number of rvr is doubled. The arena is similar to the simulation 3 arena but the radius is increased (doubled) to allow the deployment of the 10 rvr. The rvr are place alongside the y axis with a spacing of 0.45m. The initial y position is set to -0.55m. The experiment length is set to 5000 times steps and the rmse threshold to 8.

4.2 Results

To interpret the locally produced global maps by the swarm, it is compared to a distributed global map. The map is generated by retrieving each local map and merging the occupancy data with an equal contribution of each participating swarm agent (the resulting map is illustrated in Figure 4.7(b)). The pixels of this map are set to six unique color following these statements :

- Unexplored or occupied pixels coordinate identical in all local maps in black
- Unoccupied pixels coordinate identical through all local maps in orange
- Unoccupied pixels coordinate identical through 4 local maps in Coral
- Unoccupied pixels coordinate identical through 3 local maps in Light Purple
- Unoccupied pixels coordinate identical through 2 local maps in Eggplant purple
- Unique unoccupied pixels coordinate in dark blue

The resulting map is illustrated in Figure 4.7(b). A size 3 median filter is applied to reduce the noise resulting in the final map (Figure 4.7(d)). The ease the results readability and comparison with the generated global map, a similar color gradient is applied to each defined grayscale levels range as illustrated in Figure 4.6

4.2.1 Simulation 1

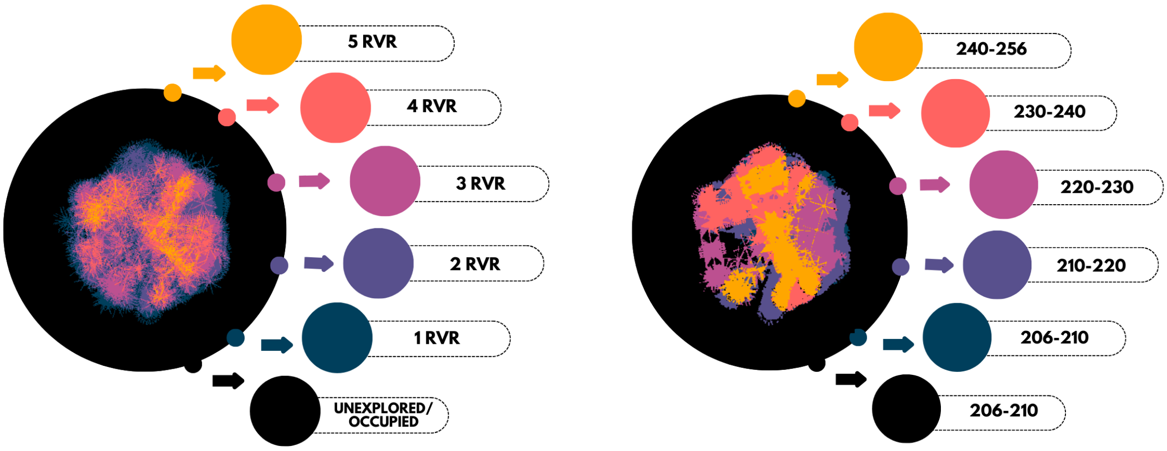
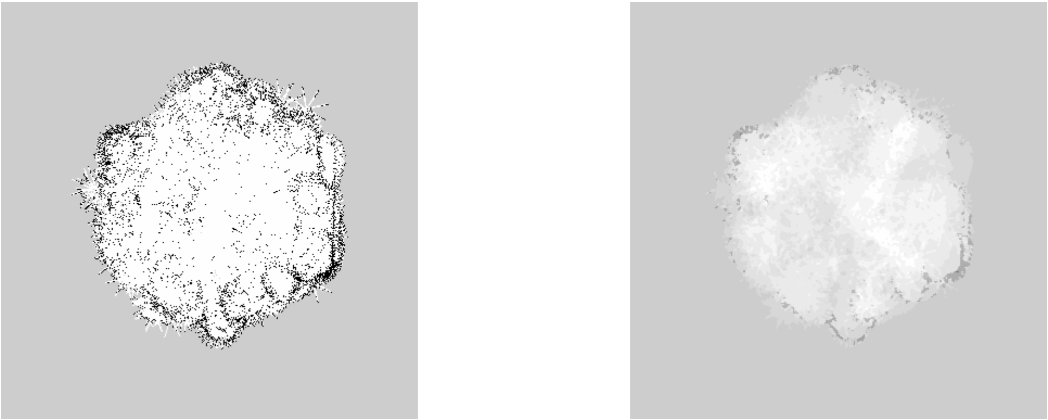
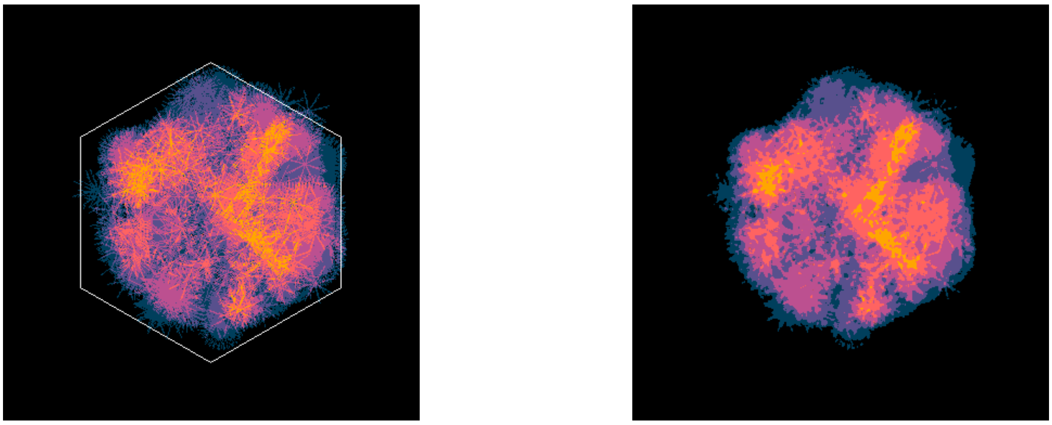

5 RVR
4 RVR
3 RVR
2 RVR
1 RVR
UNEXPLORED/ OCCUPIED
240-256
230-240
220-230
210-220
206-210
206-210

Figure 4.6: Simulation 1; Seed 7; Legended maps



(a) Merge of the individual maps; Area defined as occupied by any rvr result in a black pixel, Area defined as unoccupied by any rvr and never defined as occupied in white; Remaining area in gray

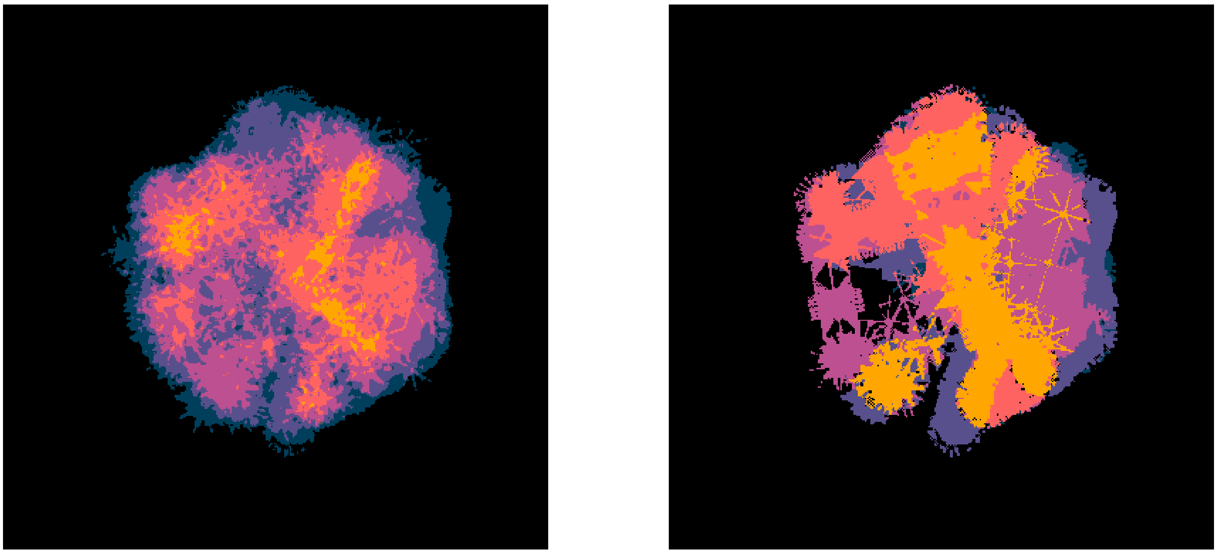
(b) Merge of the individual maps; The gray-scale level of the pixels reflect the contribution of each rvr individual maps



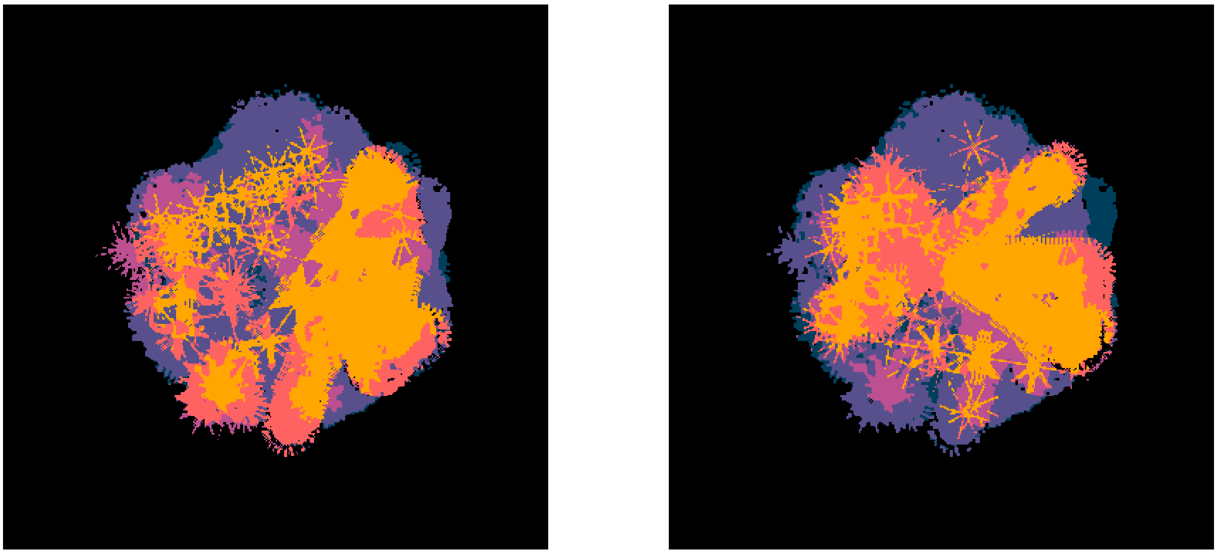
(c) Merge of the individual maps; Coloring depending on the gray-scale levels; White outline of the supposed placement of the arena

(d) Application of a size 3 median filter

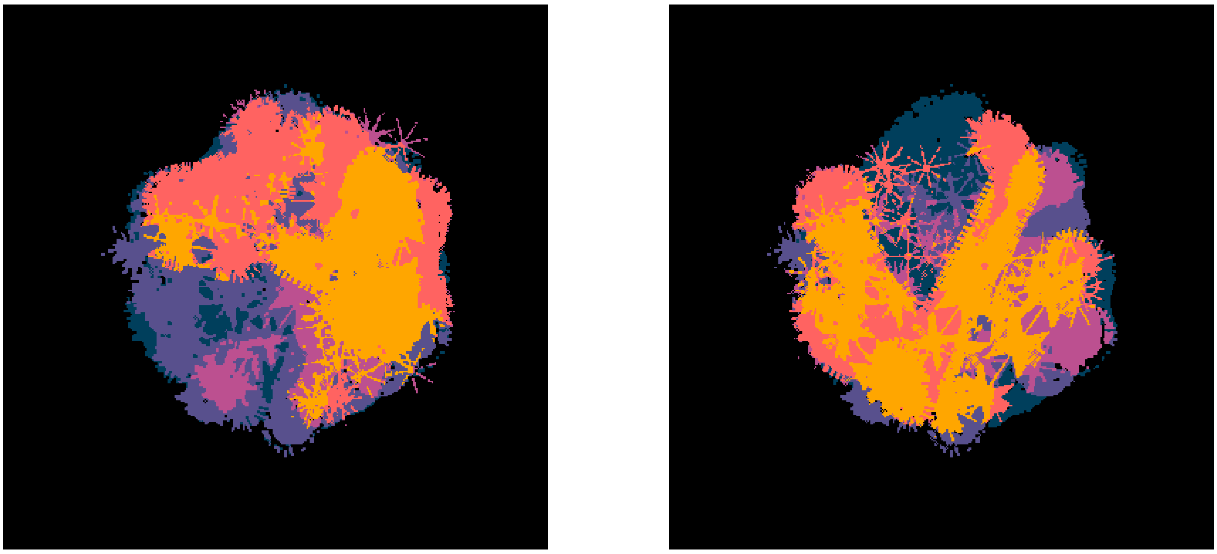
Figure 4.7: Simulation 1; Seed 7; Individual maps merged



(a) Merge of the individual maps (b) rvr0 global map



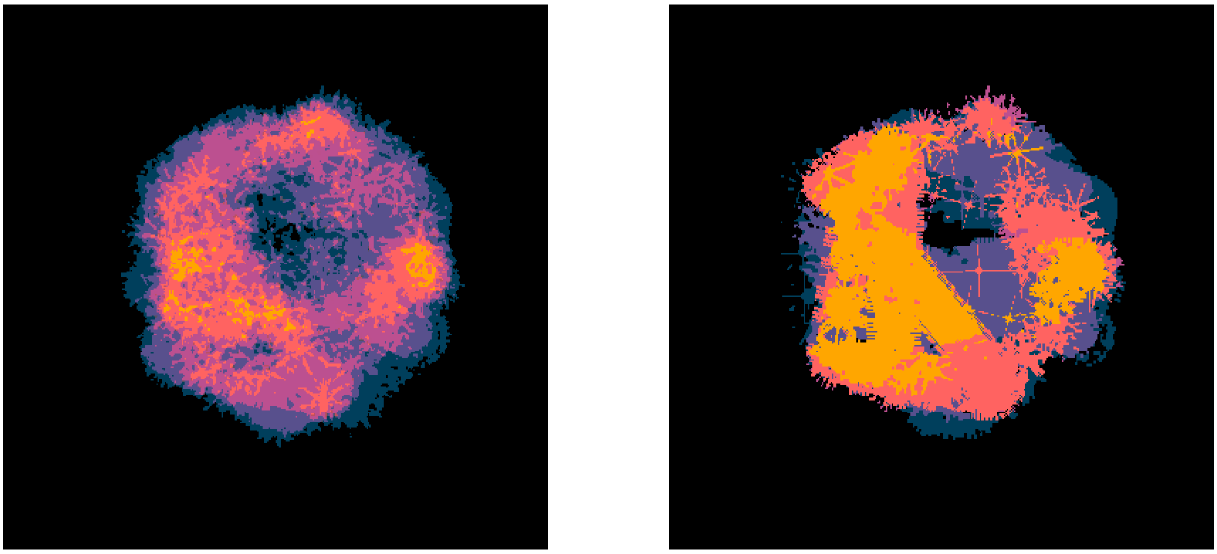
(c) rvr1 global map (d) rvr2 global map



(e) rvr3 global map (f) rvr4 global map

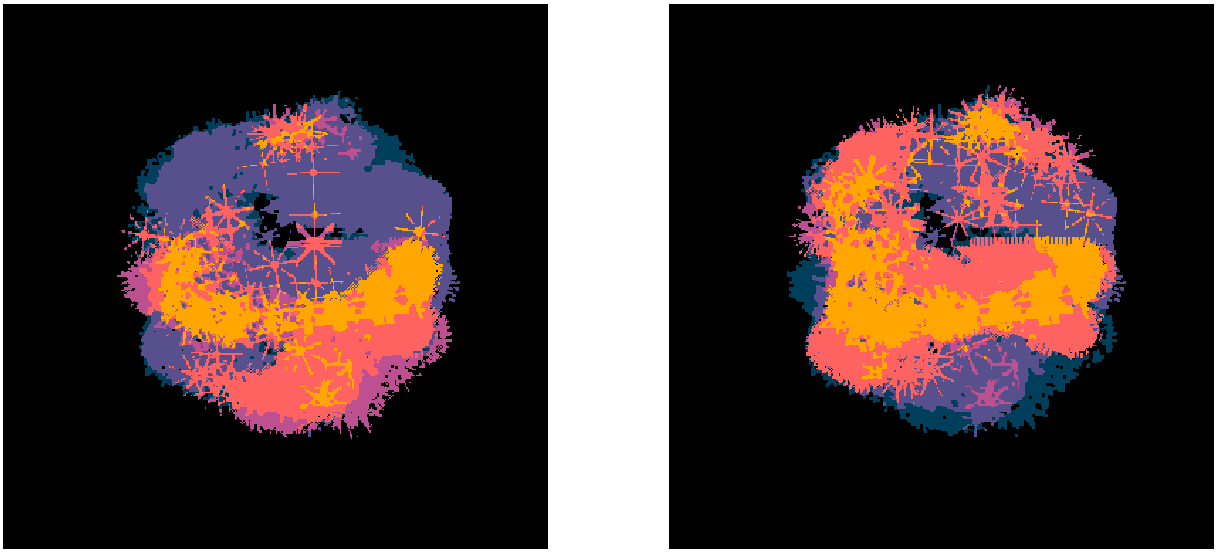
Figure 4.8: Simulation 1; Seed 7; Maps

4.2.2 Simulation 2



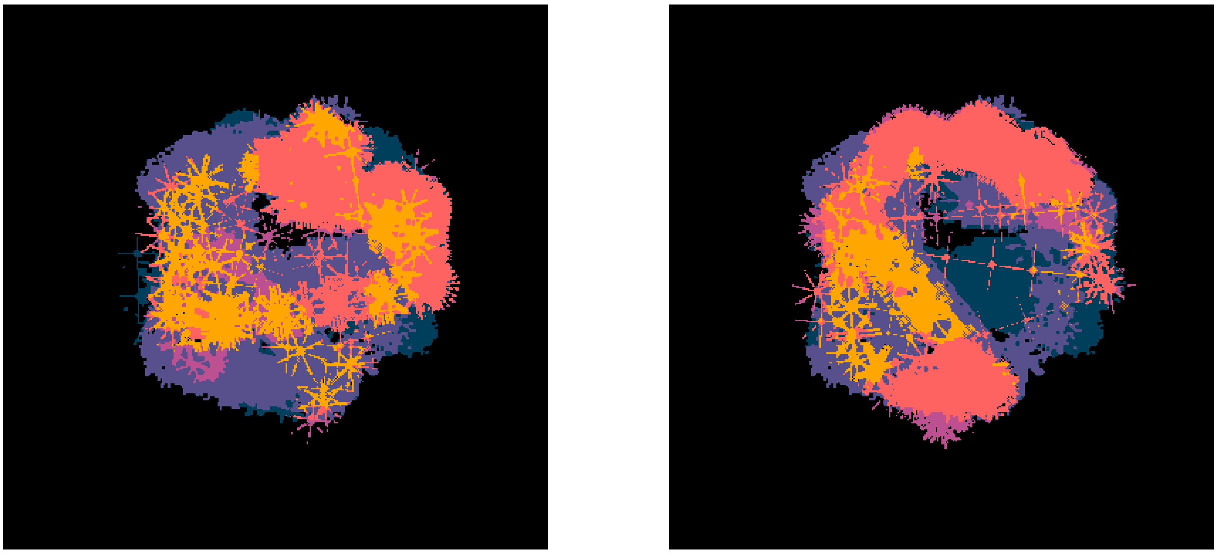
(a) Merge of the individual maps

(b) rvr0 global map



(c) rvr1 global map

(d) rvr2 global map

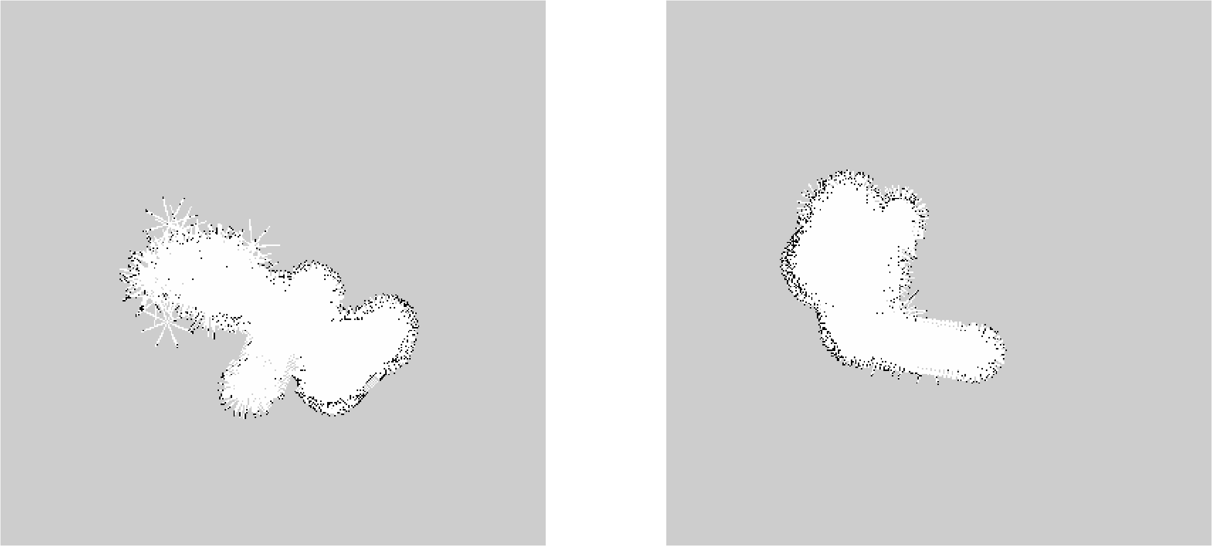


(e) rvr3 global map

(f) rvr4 global map

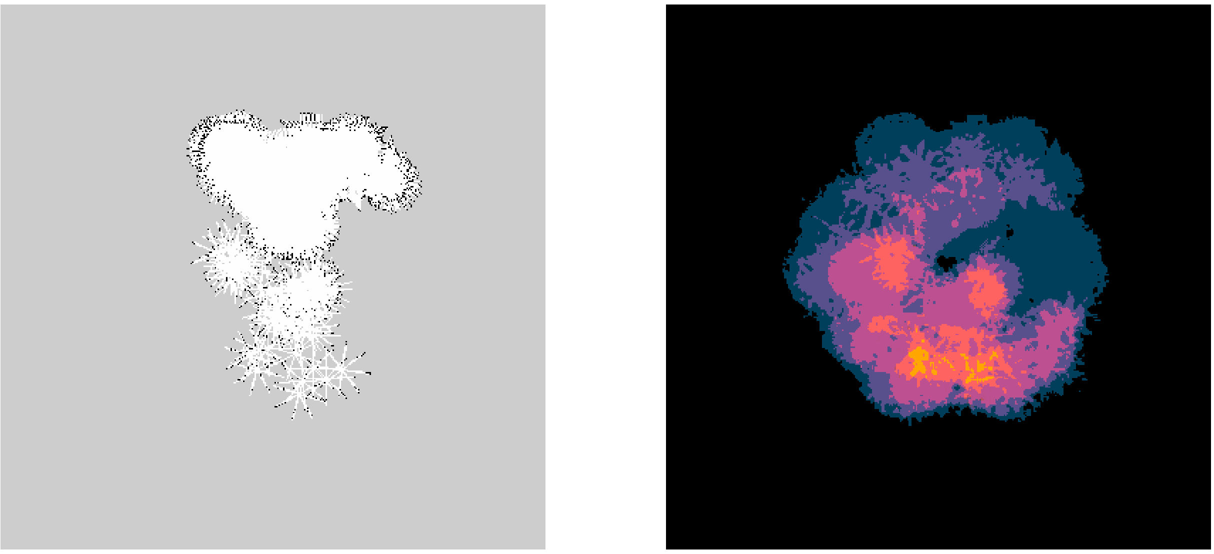
Figure 4.9: Simulation 2; Seed 9; Maps

4.2.3 Simulation 3



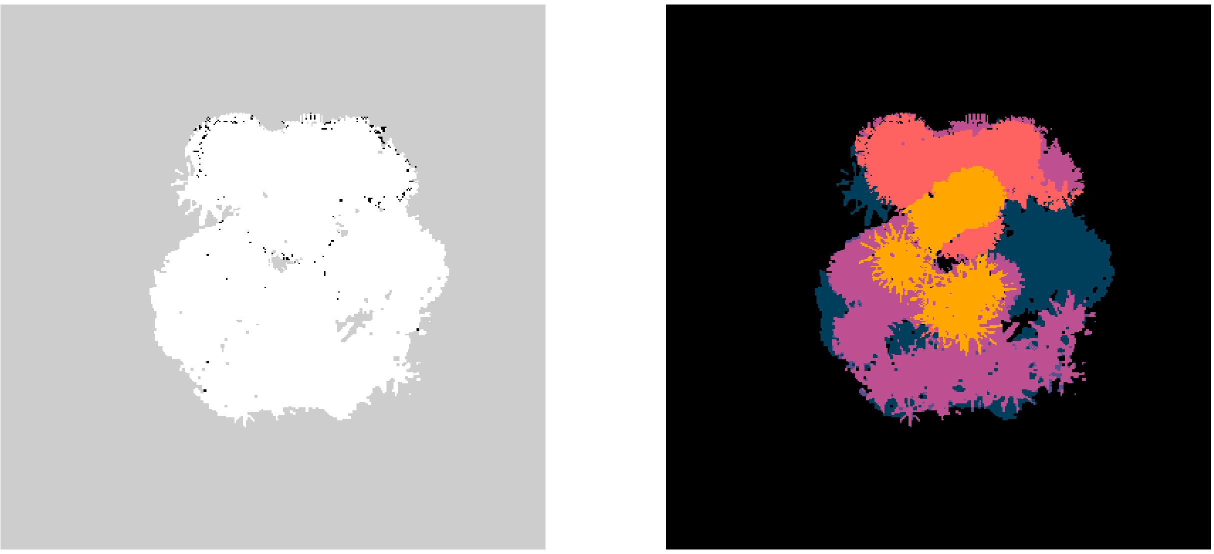
(a) rvr1 local map

(b) rvr2 local map



(c) rvr4 local map

(d) Merge of the individual maps

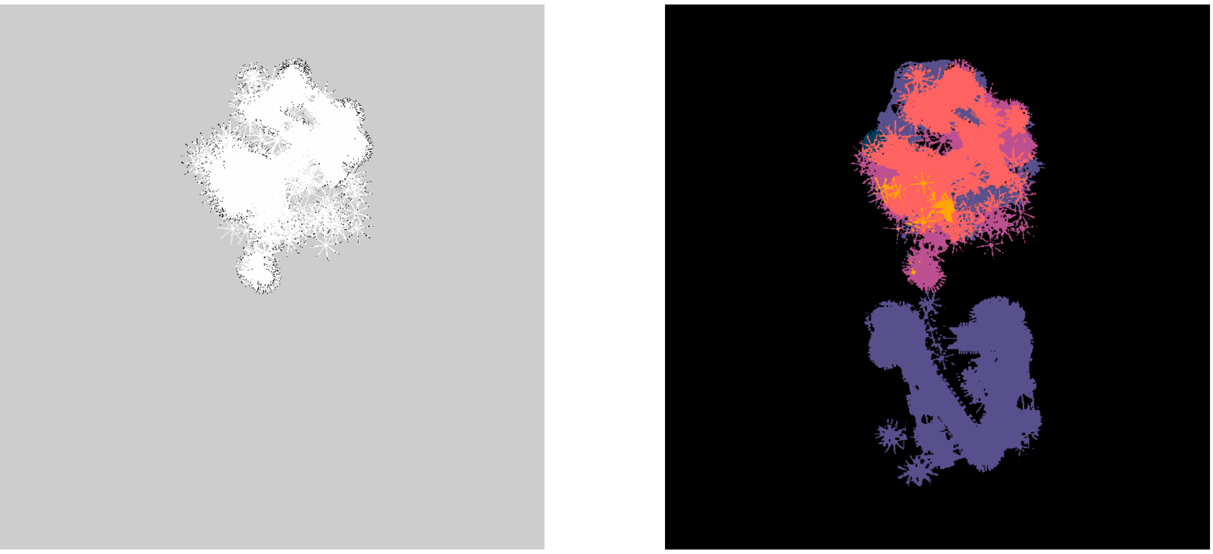


(e) rvr4 global map; Grayscale range 206-255 considered unoccupied without distinction

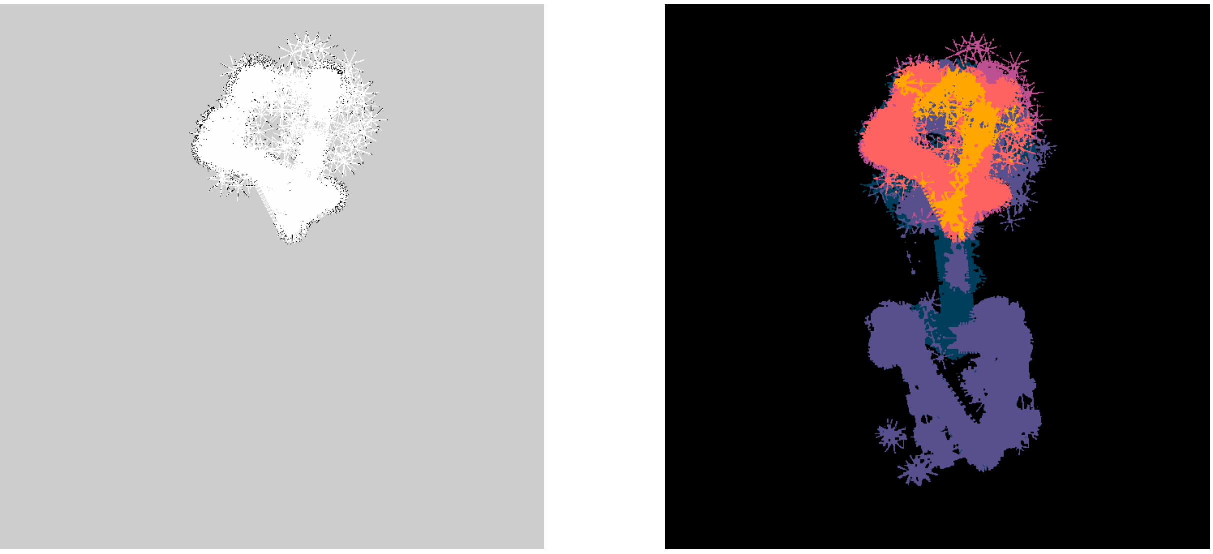
(f) rvr4 global map

Figure 4.10: Simulation 3; Seed 9; Maps

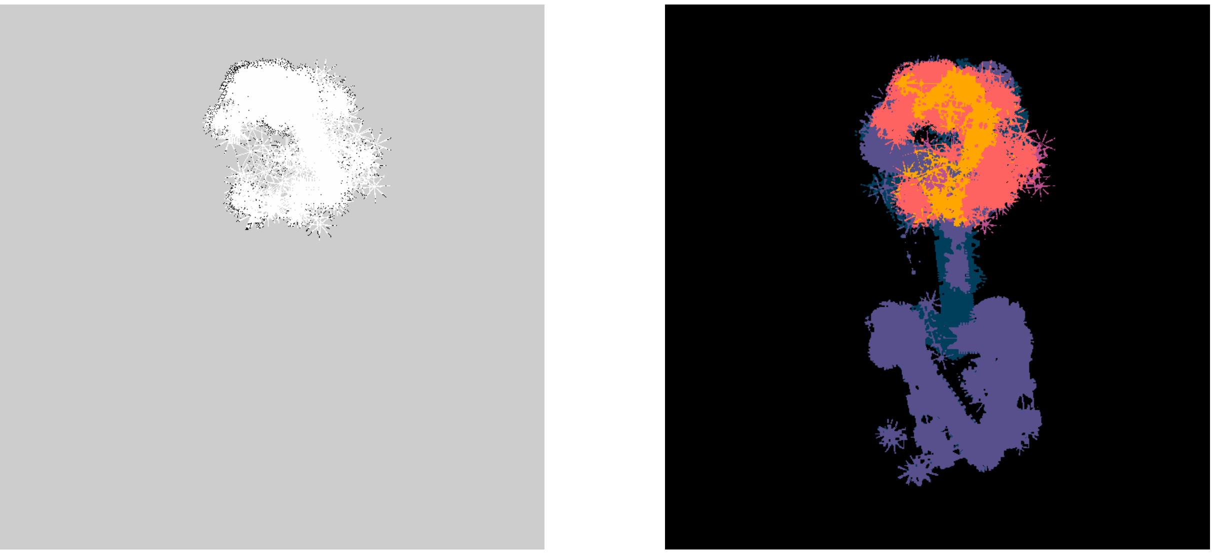
4.2.4 Simulation 4



(a) Local map rvr2 (b) Global map rvr2



(c) Local map rvr3 (d) Global map rvr3



(e) Local map rvr4 (f) Global map rvr4

Figure 4.11: Simulation 4; Seed 5; rvr2-rvr4 maps

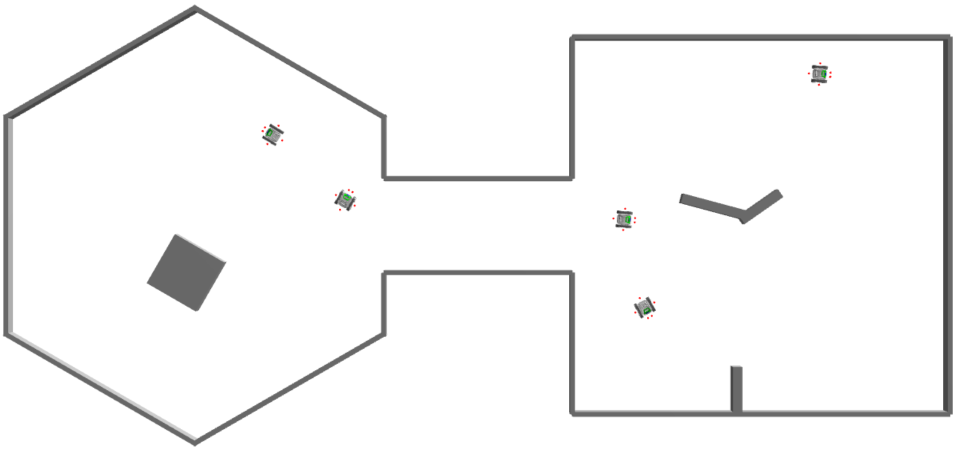


Figure 4.12: Simulation 4; Seed 1; Time frame 125

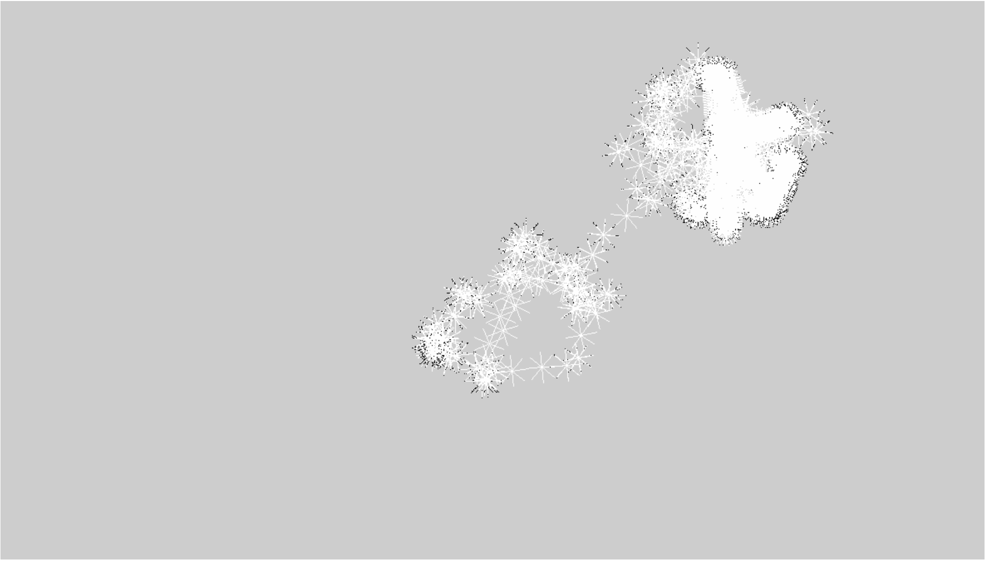


Figure 4.13: Simulation 4; Seed 3; Local map rvr3

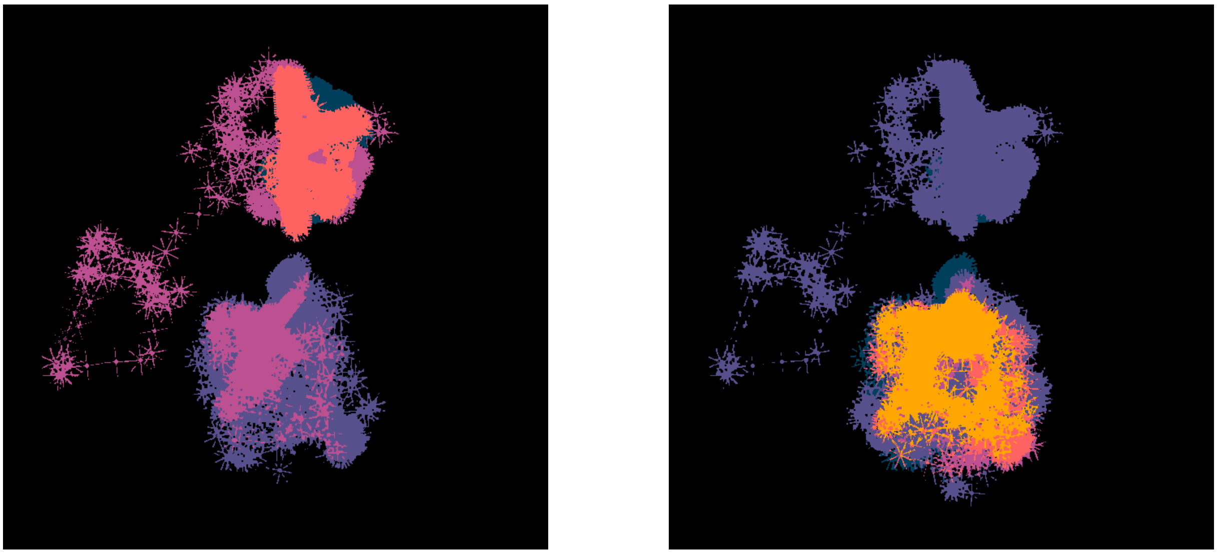


Figure 4.14: Simulation 4; Seed 3; Global map rvr3 (left) and rvr1 (right)

4.2.5 Simulation 5

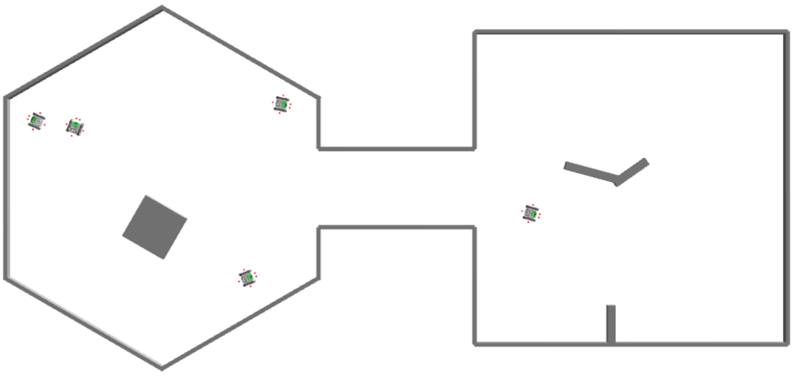
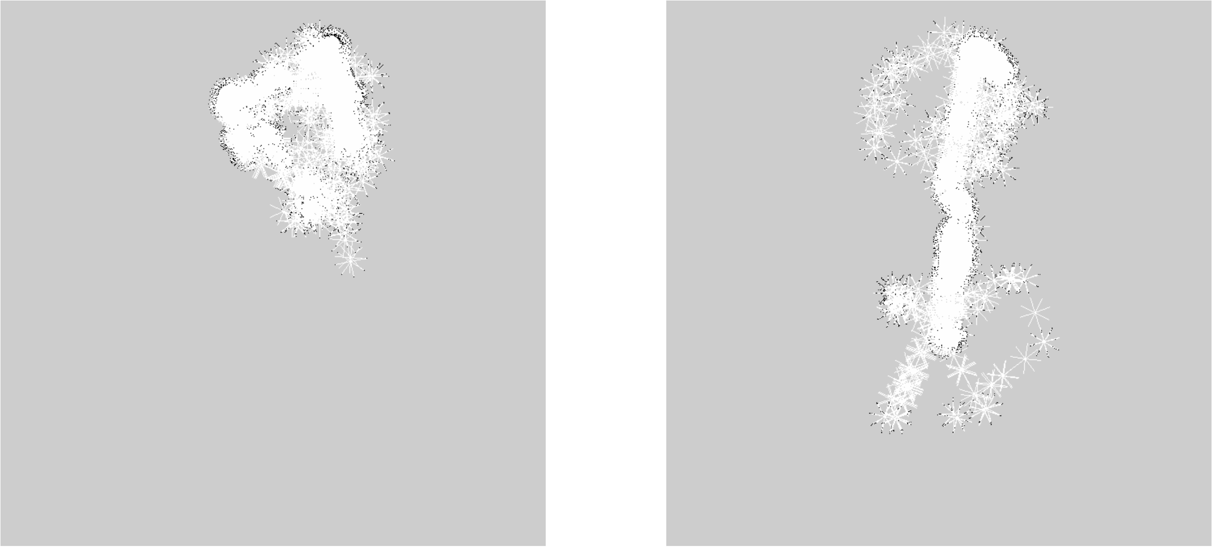
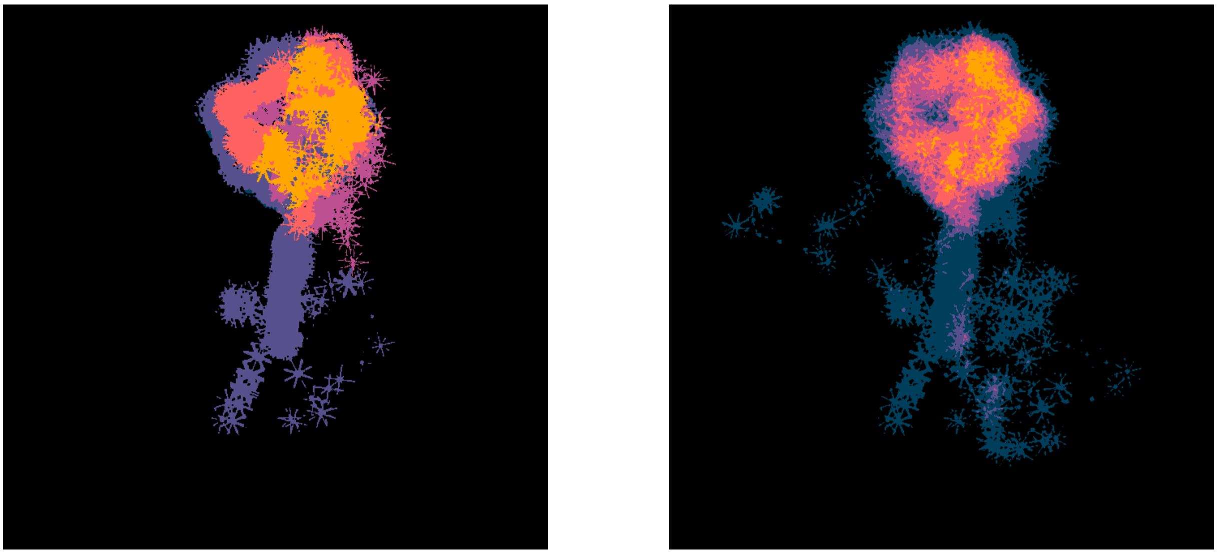


Figure 4.15: Simulation 5; Seed 1; Time frame 540



(a) Local map rvr3

(b) Local map rvr4

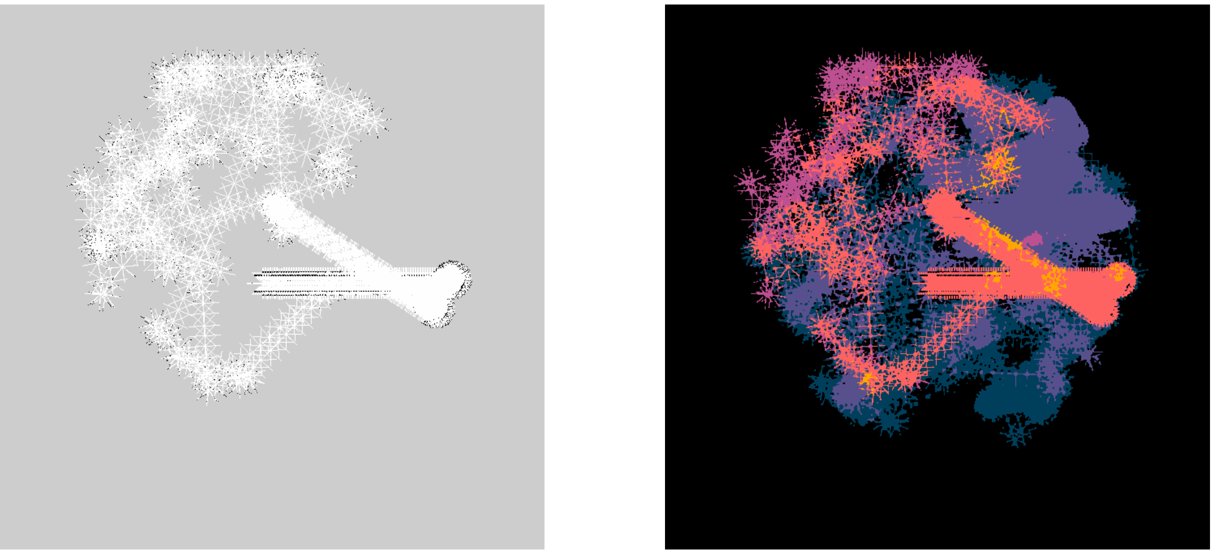


(c) Global map rvr3

(d) Merge of the individual maps

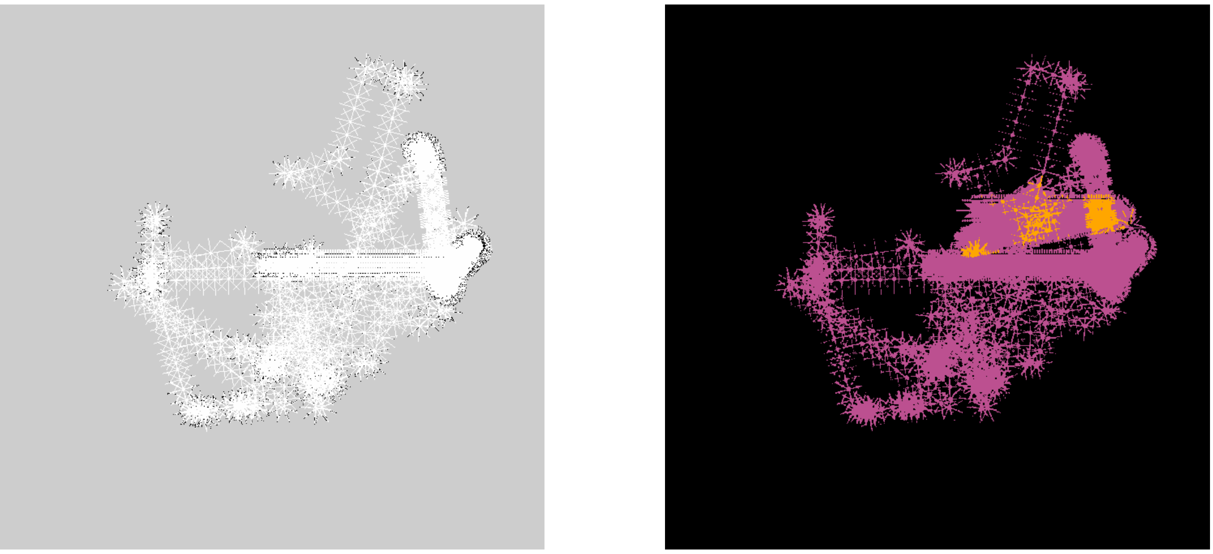
Figure 4.16: Simulation 5; Seed 8; Maps

4.2.6 Simulation 6



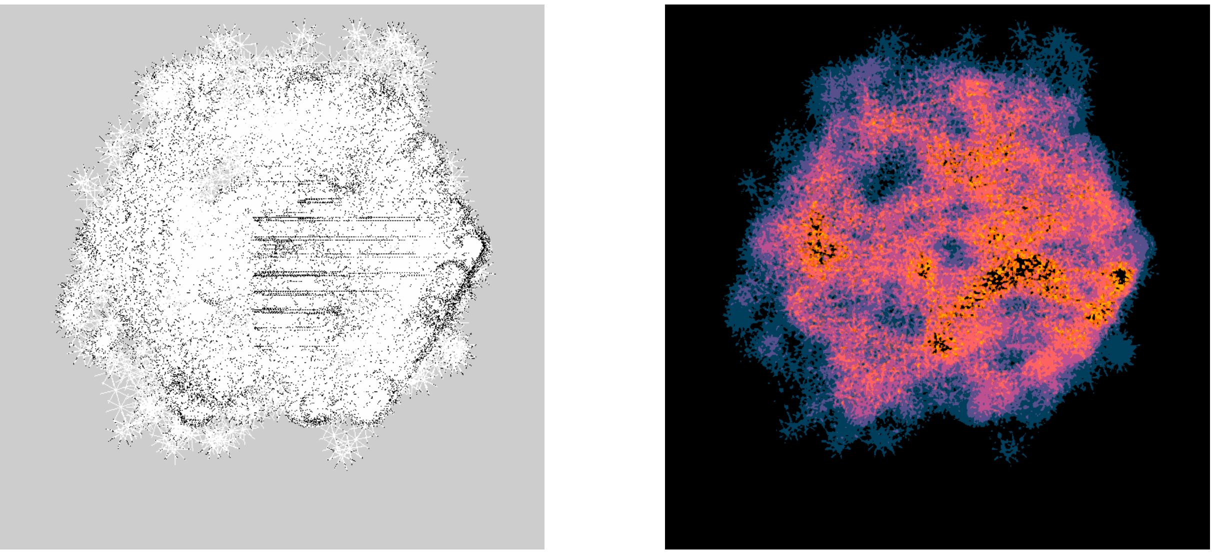
(a) Local map rvr3

(b) Global map rvr3



(c) Local map rvr4

(d) Global map rvr4



(e) Merge of the individual maps; Area defined as occupied by any rvr result in a black pixel, Area defined as unoccupied by any rvr and never defined as occupied in white; Remaining area in gray

(f) Merge of the individual maps; Coloring depending on the gray-scale levels, reflect of the contribution of each rvr individual maps

Figure 4.17: Simulation 6; Seed 7; Maps

4.3 Observation and interpretation

	Average images received	**Average images accepted**	**Percentage of images accepted on average**	**Average images rejected**	**Percentage of images rejected on average**	**Percentage of rvr without images received on average**
Simulation 1	16.64	3.78	22.72	0.48	2.88	8.00
Simulation 2	30.06	7.56	25.15	0.00	0.00	0.00
Simulation 3	35.62	3.10	8.70	6.72	18.87	4.00
Simulation 4	32.70	1.78	5.44	6.04	18.47	24.00
Simulation 5	58.00	2.00	3.45	12.66	21.83	4.00
Simulation 6	88.95	3.65	4.10	4.92	5.53	5.00

Table 4.1: Image messages exchange

	5 rvr correspondence percentage	**4 rvr correspondence percentage**	**3 rvr similarity percentage**
Simulation 1	11.58	53.00	83.00
Simulation 2	9.71	52.00	78.00
Simulation 3	3.72	29.00	67.00
Simulation 4	0.00	19.00	43.00
Simulation 5	28.40	68.00	83.00

Table 4.2: Simulation 1-5; Map correspondence percentage

	Percentage of rvr that explored both rooms	**Percentage of rvr that received message(s) about unexplored room**
Simulation 4	20.00	28.00
Simulation 5	26.00	2.00

Table 4.3: Simulation 4-5; Exploration diversification

4.3.1 Simulation 1 and 2

The condition of the simulation 1 are explicitly unrealistic to observe the results in near-perfect conditions. The Figure 4.7 feature distinct global maps obtained by merging the retrieved local maps of the swam. The global map as presented in Figure 4.7(d) and detailed in section 4.2 is used for the comparison with the different rvr dynamically produced global map.

The quick visual comparison of the produced map in both simulation could attribute a better capability to define the border of the explored area in simulation 1 but the analysis of the merged map of the individual map (as illustrated in Figure 4.8 and 4.9) indicate that this is an artifact resulting by the difference of the produced local map by each swarm.

The simulation 1 and 2 differ by the value of the rmse threshold, respectively fixed at 8 and 4. The average number of images received is independent of the rmse threshold. Only the acceptance rate should be interpreted. The average percentage of images accepted in both simulation (Table 4.1; respectively 22.72% and 25.15%) is quite similar as is the average percentage of images rejected (Table 4.1; respectively 2.88% and 0.00%).

The results of Table 4.2 is interpreted in the sub-section *simulation 4 and 5*.

4.3.2 Simulation 3

The merge of the individual maps (sub-figure 4.10(d)) shows that partial subarea of the map's pixels have not been defined as occupied by 3 to 4 rvr. A small subarea at the center seem to have reach a consensus of unexploredoccupied pixels. The sub-figures 4.10(a), 4.10(b) and 4.10(c) suggest that the presence of obstacle restrained the size of the explored area, their increased need for obstacle avoidance reducing their exploration capability. It also illustrate that the separate use of the robot's local maps is not suitable for obstacle detection.

The sub-figure 4.10(e) and 4.10(f) compare the global map generated by rvr4 corresponding to two different conditions. In the sub-figure 4.10(e), all received occupancy data are processed unilaterally. Any area defined as occupied result in a black pixel, any area defined as unoccupied and never as occupied in a white pixel; the remaining pixels are gray. In the sub-figure 4.10(f), the global map is generated as described in the rvr controller and the pixels are colored according to their gray-scale ranges. The lower color ranges suggest that the occupancy data is older and has not been corroborated as frequently, suggesting less confidence over the definition of small sub-area as unoccupied. These sub-area could inform on the possibility of presence of an obstacle. The higher color levels suggest sub-area that are more likely to be unoccupied. Information that could be useful in path planning applications.

4.3.3 Simulation 4 and 5

The same arena was used for simulation 4 and 5. The simusystemlations differ by the initial position of the robots. In the first configuration, the robots are aligned along the junction section between the two rooms, improving their ability to spread across the two rooms during the initial time steps (Figure 4.12). In the second, the robots initiate their exploration in the hexagonal room, increasing the average time frame in which the first robot reach the second room (Figure 4.15).

The higher average percentage of rvr without any image message received in the first configuration could be attributed to the initial spreading of the robots across the two rooms, lowering their opportunity of interaction with another robot.

Despite the lower percentage of rvr that managed to explore areas in both rooms (Table 4.4; 20% in simulation 4 and 26% in simulation 5) , a significantly higher percentage of rvr received occupancy data concerning the two rooms (Table 4.4; 48% in simulation 4 and 28% in simulation 5). Figure 4.11 illustrate the gain of information that robots only exploring one sub-arena can gain with the interaction with rvr that explored a different sub-area.

Figure 4.16 illustrate the only occurrence in simulation 5 of a rvr that only manage to explore one room received information regarding the second half of the arsystemena.

The Figure 4.13 appear to result from mismatched odometry, increasing the map size and misalignment of occupancy data. These error are not completely handled. The map is resized to allow the integration (as shown in Figure 4.14) and communication of data. However, the orientation is never reconsidered, resulting in a misalignment of the occupancy data in the robot's global map. Similar results were observed in simulation 5.

The Table 4.2 indicate the average percentage of coordinate correspondence between the colored pixels of the global map generated by the rvr and the global map generated by merging all local map. The correspondence is computed between the orange pixels of the rvr's global map (corresponding to the gray-scale range 240-256, supposedly the higher converging occupancy data about unoccupied pixels), and three different subarea of the global map generated by merging all local map.

1. *Orange*, pixels defined as unoccupied by all 5 rvr:
   Simulation 4, **0.00%**; Simulation 5, **28.40%**
2. *Orange* subarea enlarged to the *coral* subarea, pixels defined as unoccupied by 4 or 5 rvr:
   Simulation 4, **19.00%**; Simulation 5, **68.00%**
3. *Orange* subarea enlarged to the *coral and light purple* subarea, pixels defined as unoccupied by 3, 4 or 5 rvr:
   Simulation 4, **43.00%**; Simulation 5, **83.00%**

This illustrate a higher diversity of occupancy data when the robots are initialized in the first configuration (simulation 4, rvr aligned in the joining section) but a higher transmission of more consistent occupancy data in the second configuration (simulation5, rvr initialized in the same room). Enlarging the swarm of robots performing the mission should improve the diversity of occupancy data and gathering of available data for both configuration.

Similar observation can be made about the simulation 1 and 2. The diminished threshold in simulation 2 slightly increased the gain and diversity of information at the expense of the consistency of occupancy data through the swarm's global map. As the gain is not significant, the chose has been made to prioritize the consistency of the data and limit the unnecessary image processing caused by the integrating of small variation of data.

4.3.4 Simulation 6

Similarities can be distinguished from the results of simulation 3 and 6. Addressing the question of obstacle by the only observation of local maps (Figure 4.17(a), 4.17(b)) is not possible. The colored merge of the individual maps (Figure 4.17(f)) provides visible information regarding potential presence of obstacles. A similar hypothesis can be made on the observation of the rvr's generated global map (Figure 4.17(b)) at the condition that the enough occupancy data were obtained (as in illustrated by Figure 4.17(c) and 4.17(d)).

Considering the size of the arena, the duration of the duration of the experiment has been increase by 300 times steps. The increased size also increased the sub-areas where obstacle avoidance in not necessary. This allowed the robots to explore a larger proportion of the arena.

The increased arena size and number of robots have led to a similar increase of inconsistency in the generated local map (similar to the one illustrated by Figure 4.13). The proposed controller is not compliant with this unpredictability, impeding its performance above an indeterminate arena and swarm size.

The comprison between Figure 4.17(e) and 4.17(f) highlights key aspects and limitations of the controller. Each rvr has to be aware of its initial position and orientation in the defined coordinate system share through the swarm. Changing the initial position of the robots require modification of the odometry publication performed by the random walk node. This often result in aligning the robots along the x axis, separate them with equal distance and identical initial orientation. At the initialisation, the swarm is advancing in the same direction with the identical speed until an obstacle is perceived within the set threshold. This non-randomized distribution increase the proportion of occupancy data defined as occupied could result in the emergence of small non-existing occupied sub-area.

The phenomenon is observed in Figure 4.17(f) but seem absent from the Figure 4.17(b), suggesting that the dynamic use of the median filter could contribute the reduce the noise generated by the perception on other robots.

The comparison of Figure 4.17(e) and 4.17(f) showcase that the gain of information provided by the controller is made at the expense of a the loss of information regarding the occupied coordinates, especially regarding the wall forming the perimeter of the area to explore. The controller is not discerning the uncertainunexplored pixels from the occupied ones, resulting in a loss of information.

Chapter 5

Real-world experiment

5.1 Design

All refereed experiment duration is approximately 2 minutes and were performed inside the arena of 1.3m displayed in Figure 5.1 as described in the implementation section. The following section present the result of four distinct experiments.

The first two experiments are performed with 3 rvr with the initial position (-1.0;0.0) for rvr0, (0.0;0.0) for rvr2 and (0.5;0.0) for rvr3. The two experiments are performed under the same conditions with the exception of the rmse threshold, respectively set to 8 and 4. The third experiment was performed with a swarm of 3 rvr. The rvr0 was replaced by a distinct robot, rvr1 with the initial position (-0.5;0.0). The conditions of the experiment were identical to the first experiment. The forth experiment was also conducted under the exact same condition as the first experiment. Contrary to the third experiment, the swarm was composed of rvr0, rvr2 and rvr3.

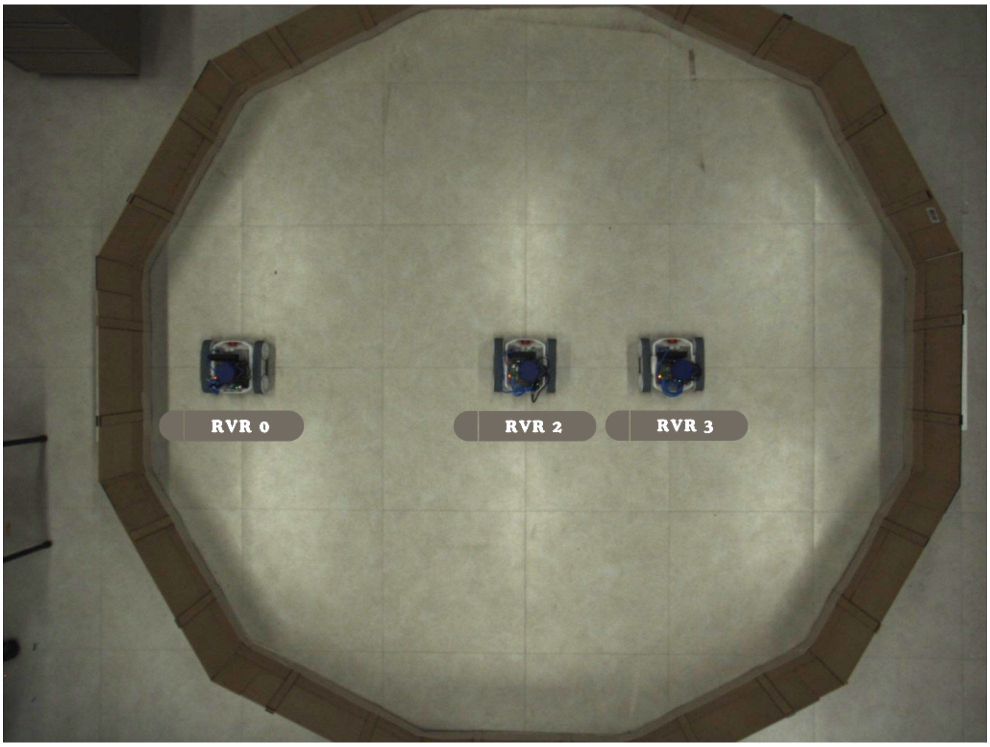


Figure 5.1: Display of the rvr inside the arena

5.2 Results

5.2.1 Experiment 1

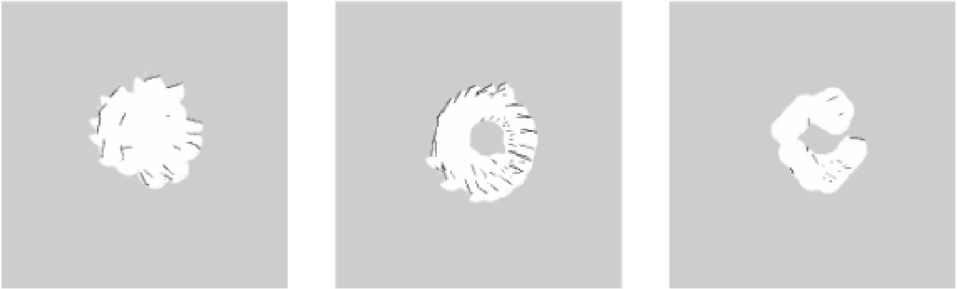
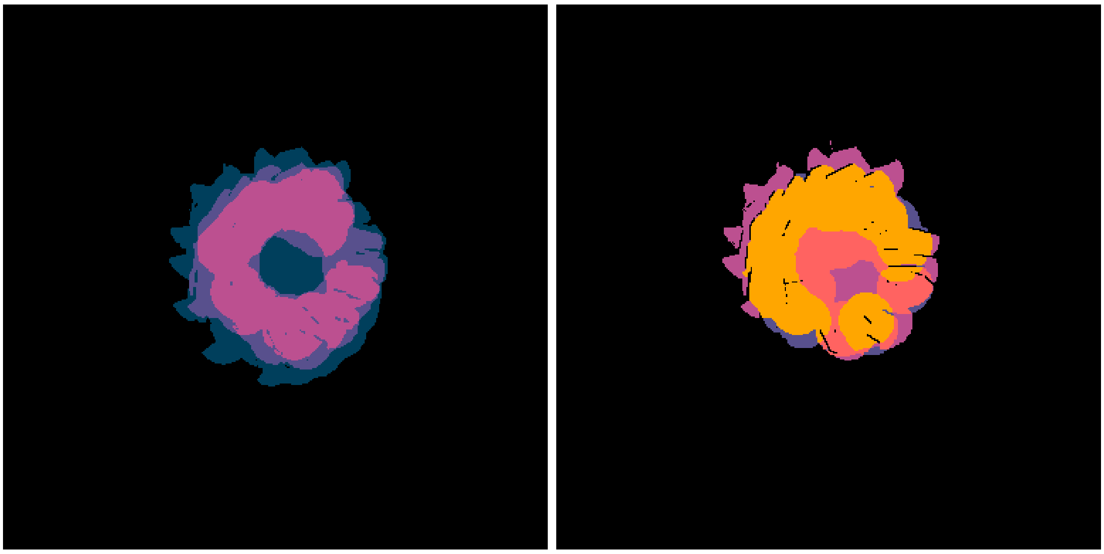
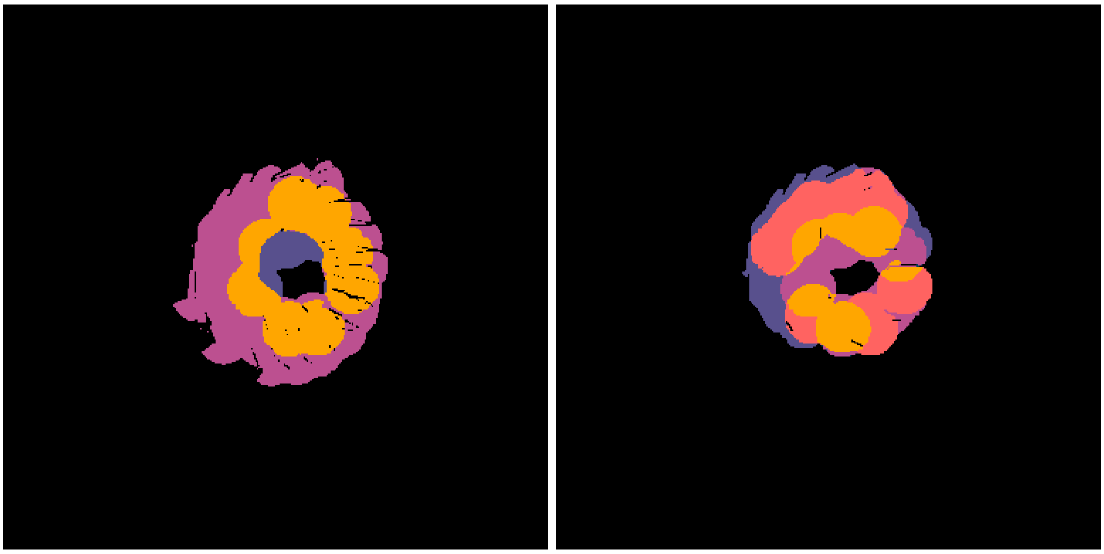


Figure 5.2: Experiment 1; Individual maps of (from left to right) rvr0, rvr2 and rvr3



(a) Merge of local maps

(b) rvr0 global map



(c) rvr2 global map

(d) rvr3 global map

Figure 5.3: Experiment 1; Global maps

5.2.2 Experiment 2

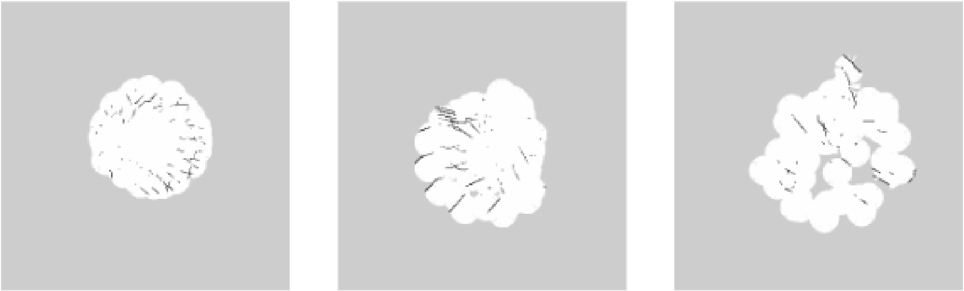
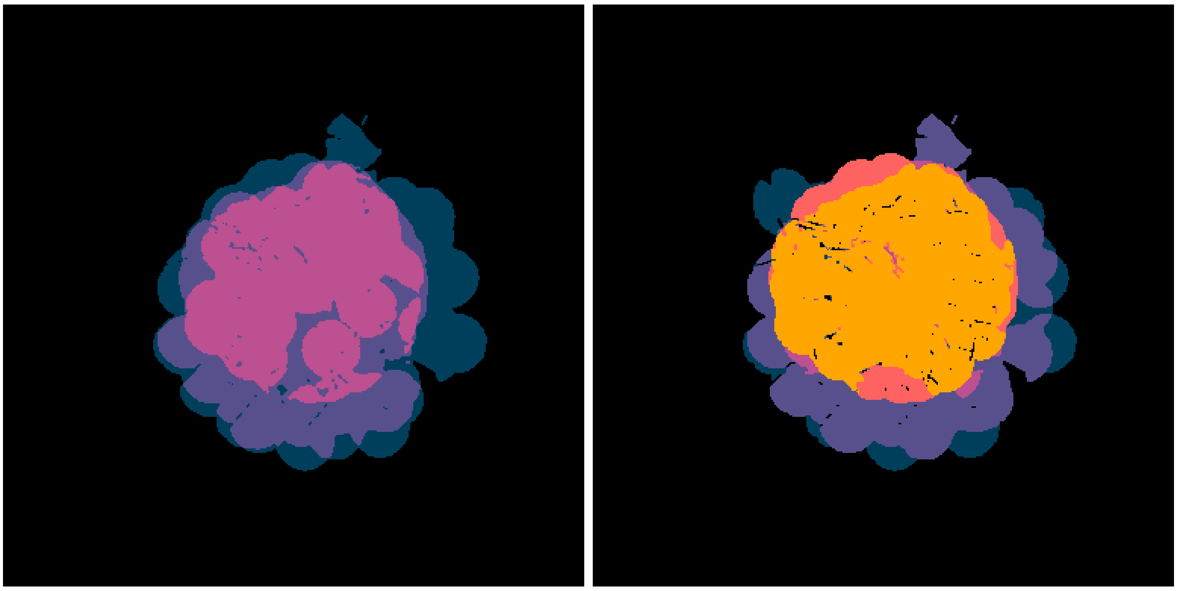
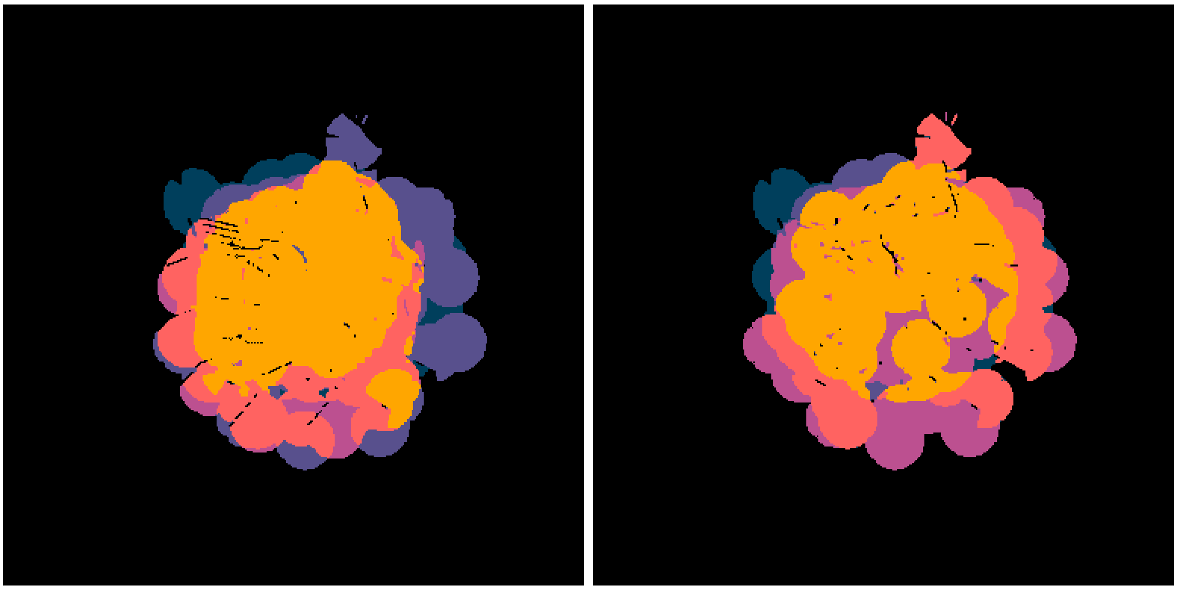


Figure 5.4: Experiment 2; Individual maps of (from left to right) rvr0, rvr2 and rvr3



(a) Merge of local maps

(b) rvr0 global map



(c) rvr2 global map

(d) rvr3 global map

Figure 5.5: Experiment 2; Global maps

	Experiment 1	Experiment 2
Rmse Threshold	8	4
Images received by rvr0	33	115
Images received by rvr2	33	115
Images received by rvr3	33	115
Images accepted by rvr0 from rvr2	1	5
Images accepted by rvr0 from rvr3	0	2
Images accepted by rvr2 from rvr0	1	1
Images accepted by rvr2 from rvr3	1	3
Images accepted by rvr3 from rvr0	0	0
Images accepted by rvr3 from rvr2	1	16
Images refused by rvr0	0	0
Images refused by rvr2	25	0
Images refused by rvr3	13	2

Table 5.1: Experiment 1-2, Map messages exchange analysis

5.2.3 Experiment 3

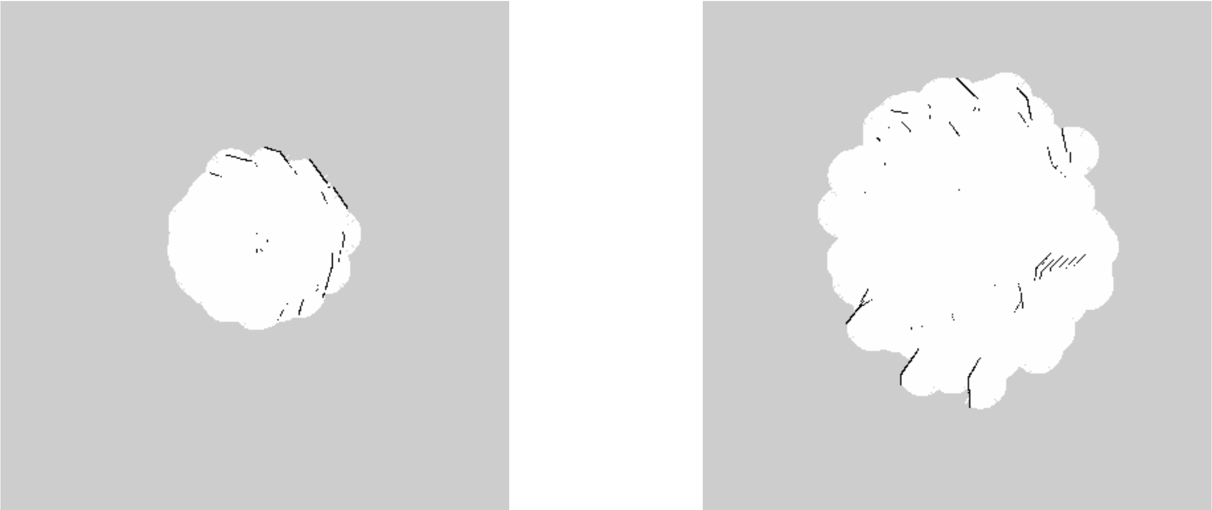


Figure 5.6: Experiment 3; Individual map of rvr2 (left) and rvr3 (right)

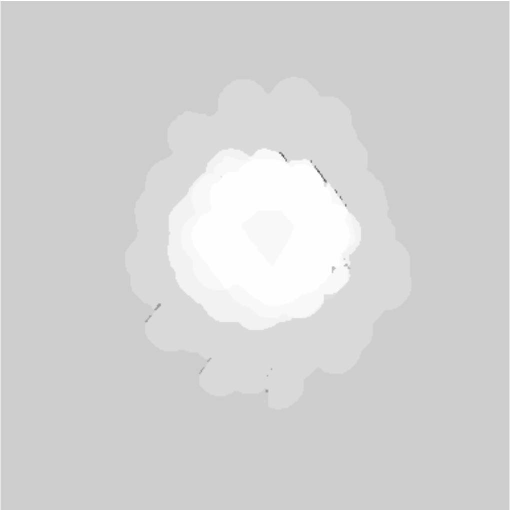


Figure 5.7: Experiment 3; Map received by rvr1

5.2.4 Experiment 4

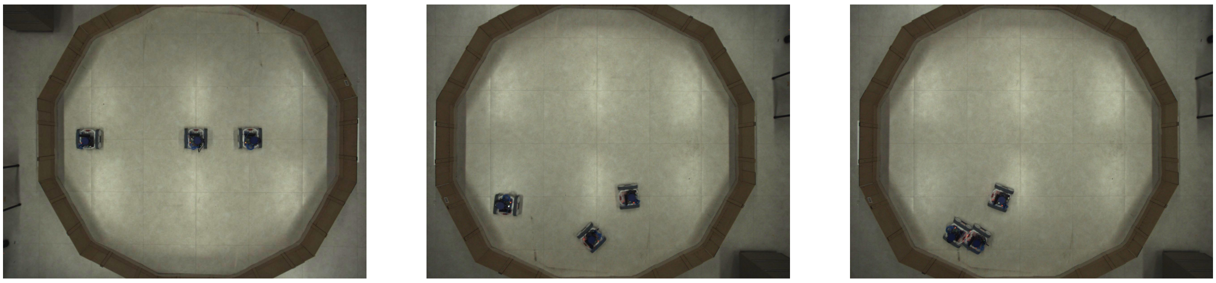


Figure 5.8: Experiment 4; Successive frames of experiment 4

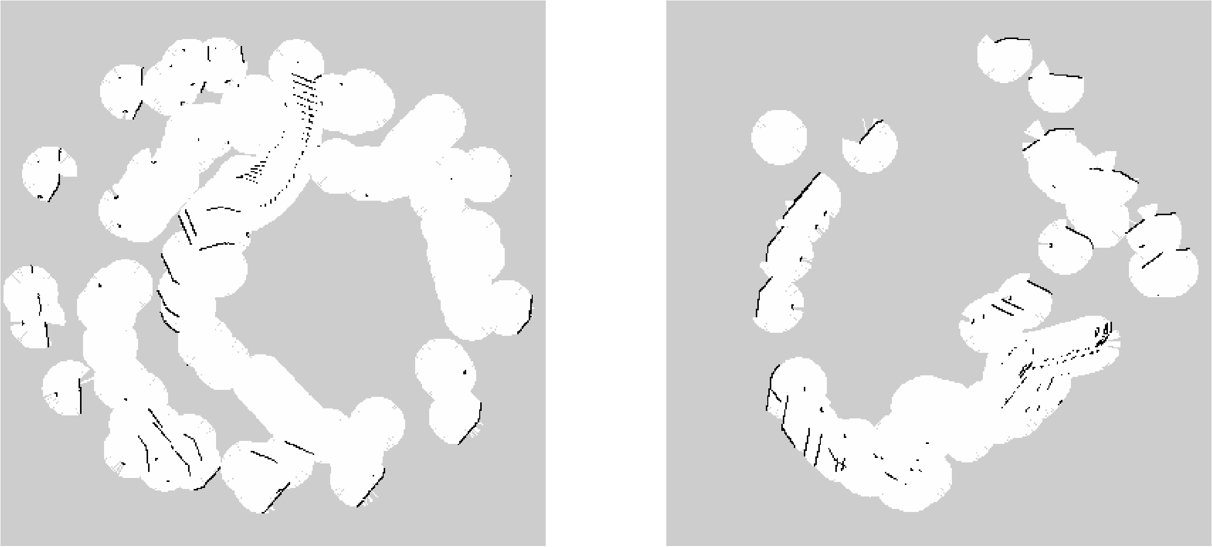


Figure 5.9: Experiment 4; Local map of rvr2 (left) and rvr3 (right)

5.3 Observation and interpretation

All refereed map messages exchanges occurring through the first two experiment are listed in Table 5.1.

5.3.1 Experiment 1

The majority of map messages were refused because the rmse threshold of 8 was not met. Rvr3 received an unique map message from rvr2 but seems to have obtain information only available in the local map of rvr0. This is only deduced from a visual observation and not measured indicating the complexity of interpretation of produced maps.

Rvr0 received an unique map message from rvr2. Whereas the distinct contribution of rvr 2 and rvr3 is less perceivable than is the distinction contribution of rvr2 and rvr0, the gradient present at the center of its global map suggest that it obtained occupancy data of rvr3 from its encounter with rvr2.

5.3.2 Experiment 2

Rvr3 was the only robot that did not received map message from both remaining robots. Its also the only robot that refused map images. However, the refused map message were from the same robot, from which it received a total of 16 map messages. The two messages were refused consecutively at about a third of the experiment length. That infer that the loss of information is negligible.

Rvr0 received 16 map messages from rvr0, some of which were exchanged over a small period of time, suggesting the redundancy of the received occupancy data.

The visual observation of the rvr0 global map suggest the presence of a similar sub-area division to the one emerging from the global merge of local images. The coordinate correspondence between the merge of local image's light purple pixels and rvr0 global image's orange pixels is 86.19% while the correspondence between the eggplant pixels is 56.50%. It separate contribution of the lowering of the threshold and the alternations of message received from both remaining rvr to these correspondences is difficult to distinguish and could benefit from retaining data that is not relevant for the accomplishment of the mission but allow a better comprehension of the contributing factors.

Rvr0 accepted a total of 7 map messages in the following order: rvr2, rvr2, rvr3, rvr2, rvr2, rvr3, rvr2.

1. **rvr2**
2. **rvr2**; *Rmse 6.67*
3. **rvr3**; *Rmse 8.33*
4. **rvr2**; *Rmse 7.54*
5. **rvr2**; *Rmse 5.00*
6. **rvr3**; *Rmse 8.74*
7. **rvr2**; *Rmse 7.37*

The gathering of the first map message occurred during the initial time steps of the experiment. The occupancy data obtained can be considerate as having a non significant contribution to the construction of the global map. The third and sixth map messages received would have been the only accepted messages for an rmse threshold set to 8. Both messages were received from rvr3. Considering that, rvr3 only received map messages from rvr0, lowering the rmse threshold significantly impacted the source diversity of the retrieved occupancy data. The rmse between the local maps of rvr0 and rvr2 is 0.20. This suggest a lower average utility of the received messages from rvr2 than rvr3 and that the loss of information due to a higher threshold could be a justifiable choice to limit the data processing of the robot's images node.

5.3.3 Experiment 3

During this experiment, there was an issue with the node *slam_gmapping0* of rvr1. The process died at the beginning of the experiment and was not able to be launched properly at any time. This means that it was not able to sent any image messages. However, it received messages from one of the rvr (rvr2) and was able to obtain a map of its own, containing information gathered by other robot(s). This indicate that the controller could benefit from the ability to create map message only from received map when the local map is missing.

5.3.4 Experiment 4

Two robots failed to avoid each other during the experiment, their wheels facing each other, prohibiting the robots to peruse their course. As the robot continue to publish the wheels_speed topic to perform their obstacle avoidance protocol, the rvr locator coordinate system seem to update the x and y coordinate even though the rvr does not manage to move. The explored area by each rvr at the time of the incident does not correspond to the produced local maps due to this odometry mismatch.

Part IV

Discussion

The incapacity of the system to perform without initial known position and orientation seem to restraint its scalability. This could be improved by integrating existing map alignment 2D grid image-based method. Applying transformation through the Hough Transform method could resolve the merging problem. [11]

The system is artificially limited by the computation of the distance between the swarm agent and the agent emitting the received message. The discussed occasional odometry mismatch impact the artificial limitation of the system as well as its mapping capability.

Currently, only the high-resolution color camera of the OAK-D camera is exploited. The exploitation of the stereo depth camera could greatly contribute to the improvement of the odometry. The potential additional resources provided by visual odometry and inter-agent range measurements could provide good localization estimates, increasing its mapping capability. Improving the robots overall localization can be made by adopting Cooperative Localization (CL) solution [61] [47].

The result of the sixth simulation brought forward the apparent noise reduction gained by using a median filer. The combined use of the median filter and the lack of distinction between occupancy data defined as unknown or occupied result in a loss of information on the border of the explored environment. Tuning the median filter parameter to remove less noise and integrating a gradient that allow a better distinction between occupied and unknown data should reduce this problematic.

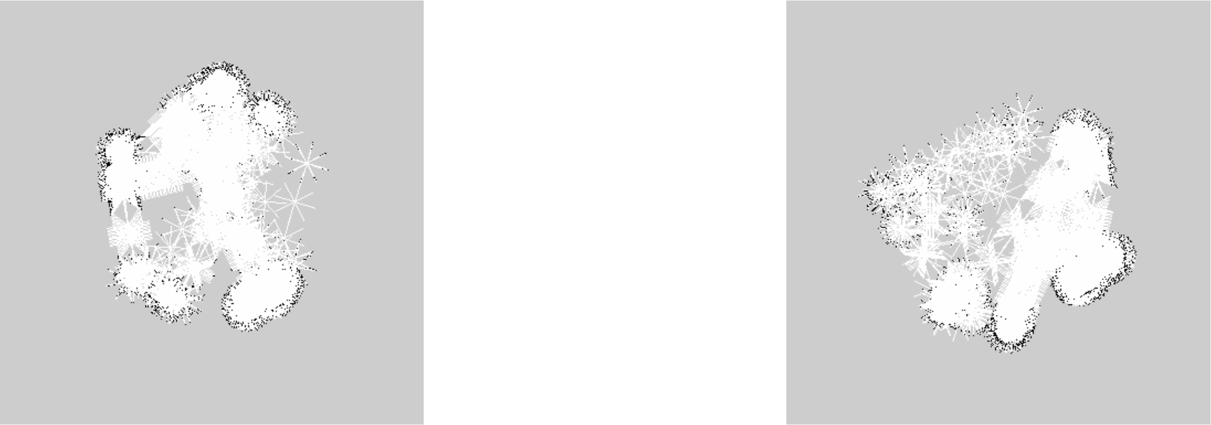
Conclusion

The proposed swarm SLAM adaptation allow the swarm to locally and dynamically create its own global map of the environment and performing its transmission while being able to retain information about the occurrence of similar received occupancy data. The limitation of exchanged and processed image messages as well as the local and dynamical processing and exchange of the occupancy data assure that the system remain decentralized and sparse.

The obtained results have demonstrated the potential scalability and flexibility of the system. While the flexibility of the system under similar conditions and diverse environment is encouraging, the capacity of the system to handle an increased environment and swarm size is not consistent, endangering its scalability.

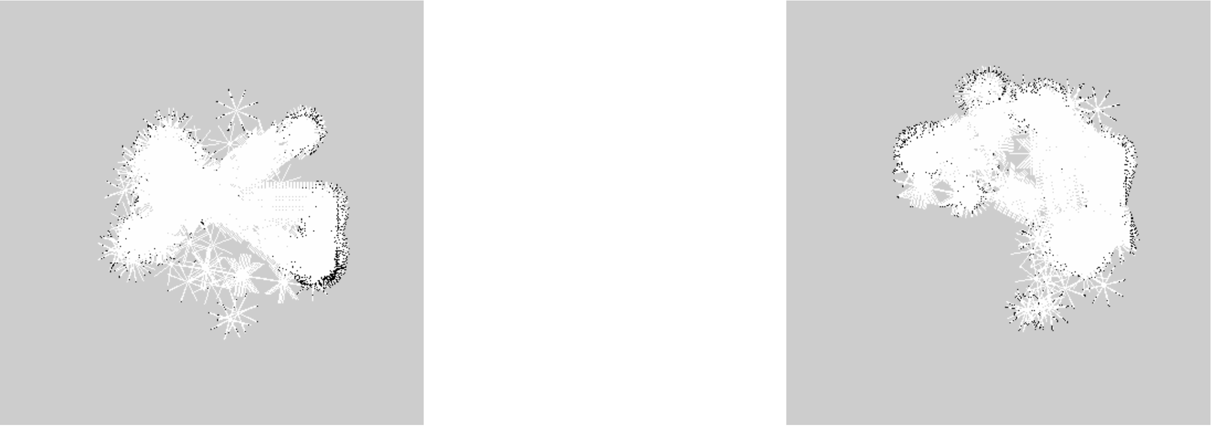
Both simulated and real-world experiment highlighted the ability of the proposed system to produce exploitable global maps as well as its limitation. The plausible possibility of integration of merging protocol allowing its possible implementation with unknown initial position and orientation increase the potential scalability of the proposed system.

Supplementary material



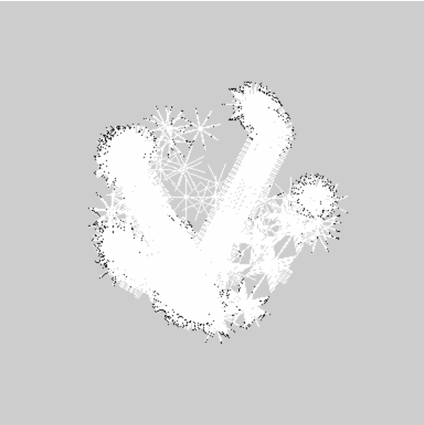
(a) rvr0 local map

(b) rvr1 local map



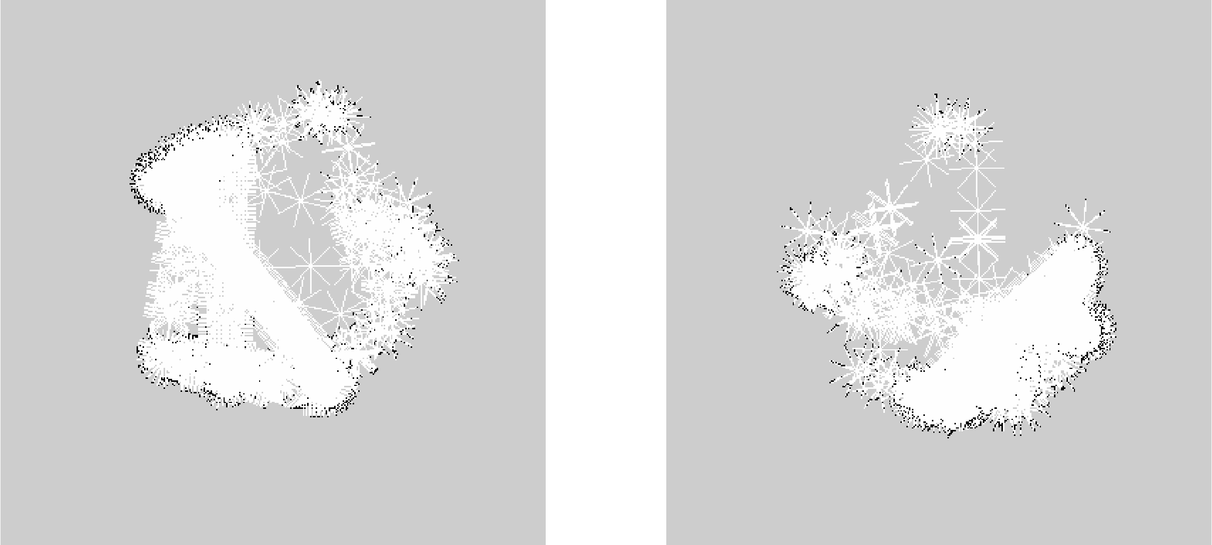
(c) rvr2 local map

(d) rvr3 local map



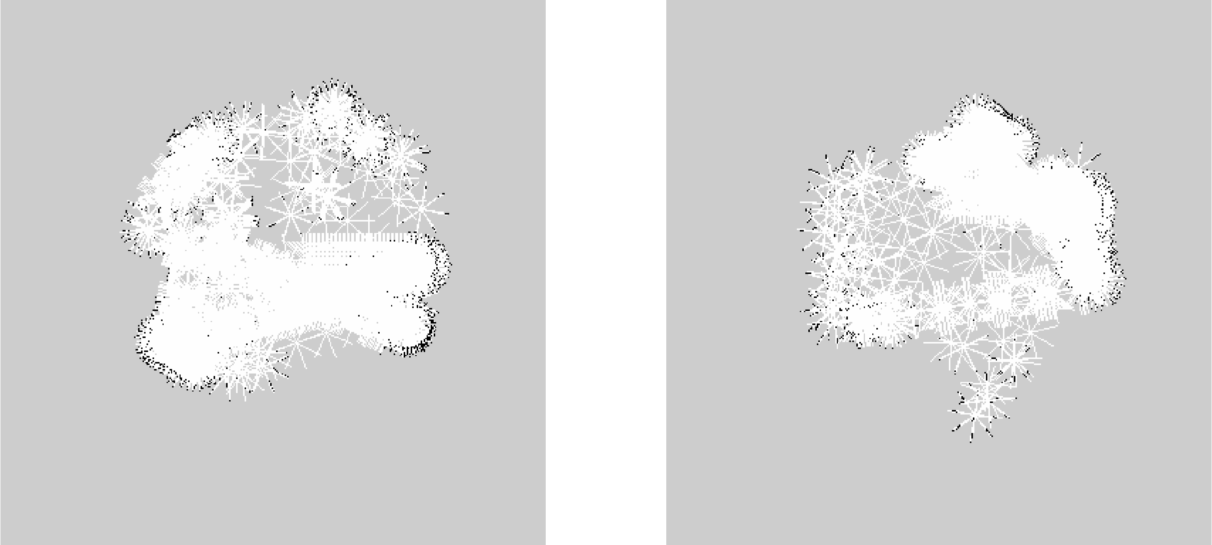
(e) rvr4 local map

Figure 5.10: Simulation 1; Seed 7; Local maps



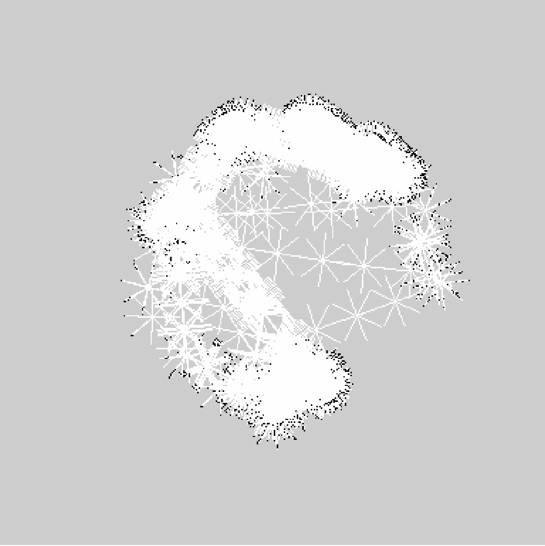
(a) rvr0 local map

(b) rvr1 local map



(c) rvr2 local map

(d) rvr3 local map



(e) rvr4 local map

Figure 5.11: Simulation 2; Seed 9; Local maps

	Orange	Coral	Light Purple	Eggplant	Dark Blue	Black
Map Merge	1677.10	82.02.80	11091.00	11547.90	8329.80	102918.50
Mean of each rvr individual count	12593.30	8209.30	6323.80	9482.00	3140.50	107705.10
Mean of each rvr similarity count	1505.40	1011.90	1914.60	4106.90	1667.00	102417.50
Correspondence percentage	11.58	12.25	31.21	43.31	54.39	95.13

Table 5.2: Simulation 1 pixels count

	Orange	Coral	Light Purple	Eggplant	Dark Blue	Black
Map Merge	1218.89	8105.22	10028.22	12607.78	8102.67	102975.33
Mean of each rvr individual count	10823.33	10610.78	3923.78	11288.33	4199.11	106324.44
Mean of each rvr similarity count	1059.78	1614.45	1159.45	4746.67	1877.56	102375.44
Correspondence percentage	9.71	14.78	27.00	41.86	45.21	96.32

Table 5.3: Simulation 2 pixels count

	Orange	Coral	Light Purple	Eggplant	Dark Blue	Black
Simulation 1	11.58	12.25	31.21	43.31	54.39	95.13
Simulation 2	9.71	14.78	27.00	41.86	45.21	96.32
Simulation 3	3.72	11.14	22.84	42.62	46.39	93.71
Simulation 4	0.00	6.24	18.60	39.91	42.68	90.25
Simulation 5	28.40	37.01	18.05	26.38	16.99	93.62

Table 5.4: Simulations 1-5 colors correspondence percentage

Bibliography

[1] Dario Albani, Daniele Nardi, and Vito Trianni. "Field coverage and weed mapping by UAV swarms". In: *2017 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*. Piscataway, NJ, USA: IEEE, 2017, pp. 4319–4325. DOI: `10.1109/IROS.2017.8206296`.

[2] Dario Albani et al. "Monitoring and mapping with robot swarms for agricultural applications". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Piscataway, NJ, USA: IEEE, 2017, pp. 1–6. DOI: `10.1109/AVSS.2017.8078478`.

[3] Ilze Andersone. "Quality Evaluation of the Occupancy Grids without Ground Truth Maps". In: *12th International Conference on Agents and Artificial Intelligence (ICAART 2020)*. Vol. 1. Setúbal, Portugal: Science and Technology Publications, 2020, pp. 319–326. DOI: `10.5220/0009175503190326`.

[4] Zachary Ball, Philip Odonkor, and Souma Chowdhury. "A Swarm-Intelligence Approach to Oil Spill Mapping using Unmanned Aerial Vehicles". In: *AIAA Information Systems-AIAA Infotech @ Aerospace*. Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2017. DOI: `10.2514/6.2017-1157`.

[5] Jan Bayer and Jan Faigl. "Decentralized Task Allocation in Multi-robot Exploration with Position Sharing Only". In: *International Symposium on Distributed Autonomous Robotic Systems 2021*. Berlin, Germany: Springer, 2021.

[6] Levent Bayındır. "A review of swarm robotics tasks". In: *Neurocomputing* 172 (2016), pp. 292–321. DOI: `10.1016/j.neucom.2015.05.116`.

[7] Mauro Birattari, Antoine Ligot, and Gianpiero Francesca. "AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms". In: *Automated Design of Machine Learning and Search Algorithms*. Ed. by Nelishia Pillay and Rong Qu. Natural Computing Series. Cham, Switzerland: Springer, 2021, pp. 73–90. DOI: `10.1007/978-3-030-72069-8_5`.

[8] Mauro Birattari, Antoine Ligot, and Ken Hasselmann. "Disentangling automatic and semi-automatic approaches to the optimization-based design of control software for robot swarms". In: *Nature Machine Intelligence* 2.9 (2020), pp. 494–499. DOI: `10.1038/s42256-020-0215-0`.

[9] Mauro Birattari et al. "Automatic off-line design of robot swarms: a manifesto". In: *Frontiers in Robotics and AI* 6 (2019), p. 59. DOI: `10.3389/frobt.2019.00059`.

[10] A. Birk and S. Carpin. "Merging Occupancy Grid Maps From Multiple Robots". In: *Proceedings of the IEEE* 94.7 (2006), pp. 1384–1397.

[11] Stefano Carpin. "Fast and accurate map merging for multi-robot systems". In: *Autonomous Robots* 25.3 (2008), pp. 305–316.

[12] Titus Cieslewski et al. "Map API - scalable decentralized map building for robots". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, NJ, USA: IEEE, 2015, pp. 6241–6247. DOI: `10.1109/ICRA.2015.7140075`.

[13] Alexander Cunningham, Vadim Indelman, and Frank Dellaert. "DDF-SAM 2.0: Consistent distributed smoothing and mapping". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 5220–5227. DOI: `10.1109/ICRA.2013.6631323`.

[14] Gianni A. Di Caro, Frederick Ducatelle, and Luca Maria Gambardella. "AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks". In: *European Transactions on Telecommunications* 16.5 (2005), pp. 443–455. DOI: `10.1002/ett.1062`.

[15] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. "Swarm robotics". In: *Scholarpedia* 9.1 (2014), p. 1463. DOI: `10.4249/scholarpedia.1463`.

[16] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization.* Cambridge, MA, USA: MIT Press, 2014.

[17] Renaud Dubé et al. “An online multi-robot SLAM system for 3D LiDARs”. In: *2017 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS).* Piscataway, NJ, USA: IEEE, 2017, pp. 1004–1011.

[18] Akif Durdu and Mehmet Korkmaz. “A novel map-merging technique for occupancy grid-based maps using multiple robots: a semantic approach”. In: *Turkish Journal of Electrical Engineering* 27.5 (2019), pp. 3980–3993. DOI: `10.3906/elk-1807-335`.

[19] Dieter Fox et al. “Distributed multirobot exploration and mapping”. In: *Proceedings of the IEEE* 94.7 (2006), pp. 1325–1339. DOI: `10.1109/JPROC.2006.876927`.

[20] Gabriele Francesca. “A modular approach to the automatic design of control software for robot swarms: from a novel perspective on the reality gap to AutoMoDe”. Doctoral dissertation. Université libre de Bruxelles, 2017.

[21] David Garzón Ramos and Mauro Birattari. “Automatic design of collective behaviors for robots that can display and perceive colors”. In: *Applied Sciences* 10.13 (2020), p. 4654. DOI: `10.3390/app10134654`.

[22] David Garzón Ramos et al. “The automatic off-line design of robot swarms: recent advances and perspectives”. In: *R2T2: Robotics Research for Tomorrow’s Technology.* Ed. by Giulia De Masi, Eliseo Ferrante, and Paolo Dario. 2021.

[23] Brian Gerkey and Tony Pratkanis. *map_server.* 2024. URL: `http://wiki.ros.org/map_server`.

[24] Bruno Duarte Gouveia, David Portugal, and Lino Marques. “Computation Sharing in Distributed Robotic Systems: A Case Study on SLAM”. In: *IEEE Transactions on Automation Science and Engineering* 12.2 (2015), pp. 410–422. DOI: `10.1109/TASE.2014.2357216`.

[25] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. “Improved techniques for grid mapping with Rao-Blackwellized particle filters”. In: *IEEE Transactions on Robotics* 23.1 (2007), pp. 34–46. DOI: `10.1109/TRO.2006.889486`.

[26] K. Hasselmann. “Advances in the automatic modular design of control software for robot swarms. Using neuroevolution to generate modules”. Doctoral dissertation. Université libre de Bruxelles, 2023.

[27] Ken Hasselmann and Mauro Birattari. “Modular automatic design of collective behaviors for robots endowed with local communication capabilities”. In: *PeerJ Computer Science* 6 (2020), e291. DOI: `10.7717/peerj-cs.291`.

[28] Ken Hasselmann, Frédéric Robert, and Mauro Birattari. “Automatic design of communication-based behaviors for robot swarms”. In: *Swarm Intelligence: 11th International Conference, ANTS 2018.* Ed. by Marco Dorigo et al. Vol. 11172. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2018, pp. 16–29. DOI: `10.1007/978-3-030-00533-7_2`.

[29] Ken Hasselmann et al. *Reference models for AutoMoDe.* Tech. rep. TR/IRIDIA/2018-002. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2018.

[30] Jiawei Hou and Haofei Kuang. *Fast 2D Map Matching Based on Area Graphs.* `http://arxiv.org/abs/1911.07432`. 2019.

[31] Tetsuya Idota, Edoardo Biagioni, and Kim Binsted. “Swarm Exploration of Extraterrestrial Lava Tubes with Ad-Hoc Communications”. In: *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE).* Piscataway, NJ, USA: IEEE, 2018, pp. 163–168. DOI: `10.1109/WiSEE.2018.8637325`.

[32] M. Kegeleirs. “Developing ROS-based software for the e-puck: An experiment in exploration and mapping”. Master’s thesis. Université libre de Bruxelles, 2018.

[33] M. Kegeleirs et al. “Automatic off-line design of robot swarms: exploring the transferability of control software and design methods across different platforms”. In: *ICRA 2023 Transferability in Robotics Workshop.* 2023.

[34] M. Kegeleirs et al. “Transferability in the automatic off-line design of robot swarms: from sim-to-real to embodiment and design-method transfer across different platforms”. In: *IEEE Robotics and Automation Letters* 9.3 (2024), pp. 2758–2765.

[35] Miquel Kegeleirs, David Garzón Ramos, and Mauro Birattari. “Random walk exploration for swarm mapping”. In: *Towards Autonomous Robotic Systems: 20th Annual Conference, TAROS 2019*. Ed. by Kaspar Althoefer, Jelizaveta Konstantinova, and Ketao Zhang. Vol. 11650. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2019, pp. 211–222. DOI: `10.1007/978-3-030-25332-5_19`.

[36] Miquel Kegeleirs, Giorgio Grisetti, and Mauro Birattari. “Swarm SLAM: challenges and perspectives”. In: *Frontiers in Robotics and AI* 8 (2021), p. 23. DOI: `10.3389/frobt.2021.618268`.

[37] Miquel Kegeleirs et al. *Mercator: hardware and software architecture for experiments in swarm SLAM*. Tech. rep. TR/IRIDIA/2022-012. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2022.

[38] Vineet R. Khare et al. “Ad-hoc network of unmanned aerial vehicle swarms for search and destroy tasks”. In: *2008 4th International IEEE Conference Intelligent Systems*. Vol. 1. Piscataway, NJ, USA: IEEE, 2008, pp. 6-65-6–72. DOI: `10.1109/IS.2008.4670440`.

[39] Jabez Leong Kit, David Mateo, and Roland Bouffanais. “A Decentralized Mobile Computing Network for Multi-Robot Systems Operations”. In: *2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. Piscataway, NJ, USA: IEEE, 2018, pp. 309–314.

[40] J. Kuckling. “Optimization in the Automatic Modular Design of Control Software for Robot Swarms”. Doctoral dissertation. Université libre de Bruxelles, 2023.

[41] J. Kuckling. “Recent trends in robot learning and evolution for swarm robotics”. In: *Frontiers in Robotics and AI* 10 (2023), p. 1134841.

[42] Jonas Kuckling, Vincent van Pelt, and Mauro Birattari. “Automatic modular design of behavior trees for robot swarms with communication capabilities”. In: *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021*. Ed. by Pedro A. Castillo and Juan Luis Jiménez Laredo. Vol. 12694. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2021, pp. 130–145. DOI: `10.1007/978-3-030-72699-7_9`.

[43] Jonas Kuckling et al. *AutoMoDe Editor: a visualization tool for AutoMoDe*. Tech. rep. TR/IRIDIA/2021-009. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2021.

[44] Jonas Kuckling et al. *Search space for AutoMoDe-Chocolate and AutoMoDe-Maple*. Tech. rep. TR/IRIDIA/2018-012. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2018.

[45] Rainer Kümmerle et al. “G^2o: a general framework for graph optimization”. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, NJ, USA: IEEE, 2011, pp. 3607–3613. DOI: `10.1109/ICRA.2011.5979949`.

[46] Pierre-Yves Lajoie and Giovanni Beltrame. *Swarm-SLAM : Sparse Decentralized Collaborative Simultaneous Localization and Mapping Framework for Multi-Robot Systems*. `https://arxiv.org/abs/2301.06230`. 2023.

[47] Young-Hee Lee et al. “Cooperative swarm localization and mapping with inter-agent ranging”. In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. Piscataway, NJ, USA: IEEE, 2020, pp. 353–359. DOI: `10.1109/PLANS46316.2020.9110227`.

[48] Guillermo Legarda Herranz et al. *Tycho: a robust, ROS-based tracking system for robot swarms*. Tech. rep. TR/IRIDIA/2022-009. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2022.

[49] A. Ligot. “Assessing and forecasting the performance of optimization-based design methods for robot swarms. Experimental protocol & pseudo-reality predictors.” Doctoral dissertation. Université libre de Bruxelles, 2023.

[50] Antoine Ligot and Mauro Birattari. “On mimicking the effects of the reality gap with simulation-only experiments”. In: *Swarm Intelligence: 11th International Conference, ANTS 2018*. Ed. by Marco Dorigo et al. Vol. 11172. Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2018, pp. 109–122. DOI: `10.1007/978-3-030-00533-7_9`.

[51] Antoine Ligot and Mauro Birattari. “Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms”. In: *Swarm Intelligence* 14 (2020), pp. 1–24. DOI: `10.1007/s11721-019-00175-w`.

[52] Antoine Ligot et al. *AutoMoDe, NEAT, and EvoStick: implementations for the e-puck robot in ARGoS3*. Tech. rep. TR/IRIDIA/2017-002. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2017.

[53] Xuan Liu et al. “Information-centric mobile ad hoc networks and content routing: A survey”. In: *Ad Hoc Networks* 58 (2017), pp. 255–268. DOI: `10.1016/j.adhoc.2016.04.005`.

[54] You Lu et al. “Energy-efficient content retrieval in mobile cloud”. In: *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. New York, NY, USA: ACM, 2013, pp. 21–26. DOI: `10.1145/2491266.2491271`.

[55] Luxonis. *DepthAI API Documentation*. 2024. URL: `https://docs-old.luxonis.com/projects/api/en/latest/`.

[56] Huma Mahboob et al. “DCP-SLAM: Distributed Collaborative Partial Swarm SLAM for Efficient Navigation of Autonomous Robots”. In: *Sensors, published online by MDPI* 23.2 (2023), p. 1025. DOI: `10.3390/s23021025`.

[57] Nathalie Majcherczyk et al. “Distributed Data Storage and Fusion for Collective Perception in Resource-Limited Mobile Robot Swarms”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5549–5556. DOI: `10.1109/LRA.2021.3076962`.

[58] Philip Odonkor, Zachary Ball, and Souma Chowdhury. “Distributed operation of collaborating unmanned aerial vehicles for time-sensitive oil spill mapping”. In: *Swarm and Evolutionary Computation* 46 (2019), pp. 52–68. DOI: `10.1016/j.swevo.2019.01.005`.

[59] Saleh Ali K. Al-Omari and Putra Sumari. “An Overview of Mobile Ad Hoc Networks for the Existing Protocols and Applications”. In: *International Journal on Applications of Graph Theory In wireless Ad Hoc Networks And sensor Networks* 2.1 (2010), pp. 87–110. DOI: `10.5121/jgraphhoc.2010.2107`.

[60] Alexandre Pacheco, Volker Strobel, and Marco Dorigo. *A Framework for Swarm Robotics Experimentation with Pi-puck Robots and an Ethereum-based Blockchain*. Tech. rep. TR/IRIDIA/2020-001. Brussels, Belgium: IRIDIA, Université Libre de Bruxelles, 2020.

[61] Anderson G. Pires et al. “Cooperative Localization and Mapping with Robotic Swarms”. In: *Journal of Intelligent & Robotic Systems* 102.2 (2021), p. 47. DOI: `10.1007/s10846-021-01397-z`.

[62] Ragesh K. Ramachandran, Zahi Kakish, and Spring Berman. “Information correlated Lévy walk exploration and distributed mapping using a swarm of robots”. In: *IEEE Transactions on Robotics* 36.5 (2020), pp. 1422–1441. DOI: `10.1109/TRO.2020.2991612`.

[63] D. Garzón Ramos et al. *MoCA: a modular RGB color arena for swarm robotics experiments*. Technical Report TR/IRIDIA/2022-014. IRIDIA, 2022.

[64] Sajad Saeedi et al. “Multiple-Robot simultaneous localization and mapping: a review”. In: *Journal of Field Robotics* 33.1 (2016), pp. 3–46. DOI: `10.1002/rob.21620`.

[65] Sajad Saeedi et al. “Occupancy grid map merging for multiple robot simultaneous localization and mapping”. In: *International Journal of Robotics and Automation* 30 (Jan. 2015). DOI: `10.2316/Journal.206.2015.2.206-4028`.

[66] M. Salman, D. Garzón Ramos, and M. Birattari. “Automatic design of stigmergy-based behaviours for robot swarms”. In: *Communications Engineering* 3 (2024), p. 30.

[67] João Machado Santos, David Portugal, and Rui Rocha. “An evaluation of 2D SLAM techniques available in Robot Operating System”. In: *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Piscataway, NJ, USA: IEEE, 2013, pp. 1–6. DOI: `10.1109/SSRR.2013.6719348`.

[68] Melanie Schranz et al. “Swarm robotic behaviors and current applications”. In: *Frontiers in Robotics and AI* 7 (2020), p. 36. DOI: `10.3389/frobt.2020.00036`.

[69] Khilda Slyusar and Miroslav Kulich. “Framework for ad hoc network communication in multi-robot systems”. In: *Acta Polytechnica CTU Proceedings* (2016), pp. 18–27. DOI: `10.14311/APP.2016.6.0018`.

[70] Gaëtan Spaey et al. “Comparison of different exploration schemes in the automatic modular design of robot swarms”. In: *Proceedings of the Reference AI & ML Conference for Belgium, Netherlands & Luxemburg, BNAIC/BENELEARN 2019*. Ed. by Katrien Beuls et al. Vol. 2491. Aachen, Germany: CEUR Workshop Proceedings, 2019.

[71] Gaëtan Spaey et al. “Evaluation of alternative exploration schemes in the automatic modular design of robot swarms”. In: *Artificial Intelligence and Machine Learning: BNAIC 2019, BENELEARN 2019*. Ed. by Bart Bogaerts et al. Vol. 1196. Communications in Computer and Information Science. Cham, Switzerland: Springer, 2020, pp. 18–33. DOI: `10.1007/978-3-030-65154-1_2`.

[72] Y. Sun et al. “Multi-robot cooperative lidar slam for efficient mapping in urban scenes”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-1/W1-2023 (2023), pp. 473–478.

[73] Danesh Tarapore, Roderich Groß, and Klaus-Peter Zauner. “Sparse robot swarms: moving swarms to real-world applications”. In: *Frontiers in Robotics and AI* 7 (2020), p. 83. DOI: `10.3389/frobt.2020.00083`.

[74] R. Todesco. “RVR: A new robot platform for swarm robotics. The building blocks of new horizons”. Master’s thesis. Université libre de Bruxelles, 2022.

[75] Mirbek Turduev et al. “Cooperative chemical concentration map building using Decentralized Asynchronous Particle Swarm Optimization based search by mobile robots”. In: *2010 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*. Piscataway, NJ, USA: IEEE, 2010, pp. 4175–4180. DOI: `10.1109/IROS.2010.5650329`.

[76] Danial Yazdani et al. “Scaling up dynamic optimization problems: a divide-and-conquer approach”. In: *IEEE Transactions on Evolutionary Computation* 24.1 (2019), pp. 1–15. DOI: `10.1109/TEVC.2019.2902626`.

[77] YDLIDAR. *YDLidar-SDK*. `https://github.com/YDLIDAR/YDLidar-SDK`. 2024.