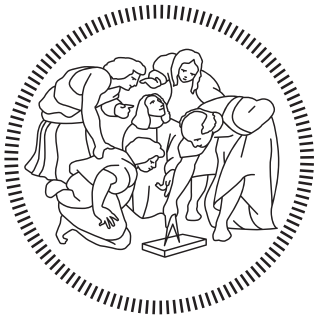
POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Dipartimento di Elettronica, Informazione e Bioingegneria
MSc in Computer Science and Engineering



Online switch of communication strategies for efficient multirobot exploration

Supervisor: Prof. Francesco Amigoni
Co-supervisors: Eng. Jacopo Banfi
Dr. Matteo Luperto

Master Thesis by:
Alessandro Longoni, ID 836599

Academic Year 2015-2016

I suggest a new strategy, R2. Let the Wookie win.
C-3PO, Star Wars - A New Hope

Abstract

Multirobot systems are formed by a number of robots which cooperate to achieve a common goal. They show flexibility in adapting to different situations and settings, like those involving the exploration of an unknown environment. In order to accomplish this task, the system requires an exploration and a coordination strategy to decide where each agent should move, as well as a communication strategy to efficiently share information between robots and a possible Base Station. All the communication strategies proposed in literature use a single communication strategy. The goal of this thesis is to develop a multirobot system for exploration that can perform an online switch of communication strategies. Specifically, we develop a system that switches, during the exploration, between two communication strategies, using an estimate on the percentage of the environment explored so far to decide when each strategy should be employed. Results show that the implemented system can reach a better trade-off, with respect to the state of the art systems, between time taken to explore the environment, distance traveled by the robots, and disconnection time with the Base Station.

Sommario

I sistemi multirobot sono formati da robot che cooperano per raggiungere un obiettivo comune. Nel corso del tempo si sono mostrati flessibili e in grado di svolgere molteplici compiti, come quello di esplorare un ambiente sconosciuto. Per fare ciò, un sistema multirobot richiede una strategia di esplorazione e coordinamento per decidere dove si devono muovere gli agenti, nonché una strategia di comunicazione per scambiare informazioni tra di loro e una possibile Base Station, la quale, per tutti i sistemi proposti in letteratura viene decisa all'inizio dell'esplorazione e mantenuta per tutta la durata L'obiettivo di questa tesi è di sviluppare un sistema multirobot che permetta il cambio online tra diverse strategie di comunicazione durante l'esplorazione di un ambiente inizialmente sconosciuto. In particolare, abbiamo sviluppato un sistema che, durante il corso dell'esplorazione, cambia tra due strategie di comunicazione, utilizzando una stima sulla percentuale dell'ambiente esplorato per decidere quando ciascuna della strategie deve essere utilizzata. I risultati mostrano che il sistema implementato ottiene un miglior bilanciamento, rispetto agli approcci dello stato dell'arte che impiegano una sola strategia di comunicazione, fra tempo impiegato per esplorare l'ambiente, distanza percorsa dai robot e tempo di disconnessione con la Base Station.

Ringraziamenti

Tanto è cambiato da quando ho messo per la prima volta piede al Politecnico, un settembre di quasi sei anni fa. Sono successe tante cose e passate tante persone che ringraziarle tutte probabilmente impiegherebbe ben più di un libro.

Desidero ringraziare il prof. Francesco Amigoni, l'ing. Jacopo Banfi e il dott. Matteo Luperto. Senza il loro fondamentale aiuto e la loro disponibilità a chiarire ogni dubbio, oggi non sarei qui.

Un ulteriore grazie va a tutti gli amici sulla mia strada, quelli di lunga data e quelli più recenti, che sono stati in grado di supportare e comprendere le mie lunghe assenze. Un grazie anche a tutti i miei compagni di università, che hanno allietato questo percorso e non hanno mai mancato il sostegno.
Un altro grazie a tutti i miei colleghi (forse sarebbe meglio dire amici) dell'AIA di Seregno. In questi nove anni ho imparato più cose di quante potessi pensare quando da sbarbato quindicenne varcavo quella porta per iscrivermi al corso arbitri.

Un grazie ancora più grande va alla mia famiglia, i miei genitori e mio fratello, perché non riesco a concepire come abbiate potuto sopportare i miei sfoghi e sbalzi d'umore da così tanto tempo. Un grazie anche ai miei nonni, anche quelli che guarderanno dall'alto questo giorno. Spero che un sorriso possa scappare anche a voi.
E grazie anche a te Cami, perché continuo a chiedermi come tu possa aver fatto a scegliere volontariamente me. Grazie per come hai sopportato il vederci poco e le sere finali chiuso in camera con te che non facevi altro che guardare e fare il tifo per me.

Più in generale, il mio ringraziamento va a tutti quelli che ho incontrato sulla mia strada anche solo per una volta o un consiglio.

Contents

List of Figures

Chapter 1

Introduction

In multirobot systems, a group of robots coordinates in order to reach a common goal. Recently, they have received special attention, given their flexibility in adapting to different situations and settings, such as exploring an unknown environment, or operating in a search and rescue setting. Sometimes, they also need to deliver collected information to a Base Station (BS). To do so, the system requires an exploration and coordination strategy to decide where the robots should move, as well as a communication strategy, which decides how and when the robots are required to connect and communicate, over a short range RF (Radio Frequency) channel, between them and the BS.

Exploration, coordination, and communication strategies in multirobot systems have been widely studied in the literature. In particular, different approaches have been proposed to cope with different communication models, connectivity constraints, and the need to deliver information to a fixed BS. A multirobot system employs only one of these two approaches: one based on a periodic connectivity constraint and one based on a continuous connectivity constraint. When dealing with a periodic connectivity constraint, we can find strategies like that of [30], which leaves robots free to explore the environment and only requires them to deliver the information to the BS when an event occurs. This kind of communication strategies belongs to the category of Rendezvous (RV) strategies. On the other side, when dealing with a (almost) continuous connectivity constraint, we have strategies like [5], in which robots maintain a connected chain with the BS throughout the course of the exploration of an unknown environment. This type of strategies belongs to the category of Multi-hop strategies (MH).

It has been observed that different communication strategies are well suited for different stages of exploration. In particular, in [1], a conjecture is advanced that a MH strategy is well suited for initial and final stages of exploration, while a RV strategy is more convenient in the middle of the exploration. The goal of this work is to empirically investigate the aforementioned hypothesis presenting an exploring multirobot system that can employ different communication strategies during an exploration mission and evaluate its efficiency when exploring an unknown environment.

In particular, we present a policy that gives the possibility to switch between different communication strategies, during the exploration of an unknown environment and without having to pause the mission. The decision on which communication strategy should be employed is based on the environment and mission status and it is decided in a centralized way by the BS. We also extend this model so it can work with two communication channels. This means that the robots not only use a short range RF channel to communicate, but they can also use in addition a second communication channel with a longer range, but a limited bandwidth (i.e, the robots and the BS cannot exchange information such as the partial maps of the environment on it). With this second channel the BS is able to communicate information in a synchronous way to all the robots (e.g., a signal to alert the robots to switch communication strategy).

We instantiate a possible implementation of our policy. Following the observations made in [1], we develop a policy that performs a switch from the starting MH strategy presented in [5] to the RV strategy explained in [30] after the initial stage of the exploration mission. Then, it switches back to the MH strategy in the final stage of the exploration mission. The decision of which strategy to employ is based on the estimate of the percentage of the total area of the environment known at the BS. This value is compared with two parameters (α and β) that represent the thresholds to change communication strategy. Since the value of the total area of the environment is usually unknown during an exploration, we utilize an estimate of it to compute the percentage of the explored environment and we compare this prediction with the thresholds to decide when the switch should take place.

We use three different models to compute this prediction. The first one uses the knowledge of the minimum bounding box of the environment to estimate its unknown area. The other two models leverage information ex-

tracted from a dataset of school buildings presented in [15]. The second model considers the mean area over the environments in the dataset and uses it as the estimate of the total area of environment the system is exploring. The third method computes the absolute error, between the real value of the area of the environment and the predicted one, made during a set of exploration runs with the method presented in [14]. Then it trains a neural network model to obtain a function that predicts this absolute error according to a given amount of explored environment. The system uses this curve to sample a value of area in a uniform interval around the value of the currently known area. The actual estimate of the total area of the environment is returned computing the median value over the last m estimates computed as before.

We test these models and the switch of communication strategies on a simulator with different environments. The experimental results show that our system with online switch of communication strategies can provide a more efficient trade-off between the time taken to explore the environment and the time in disconnection among the robots with respect to state of the art communication strategies which usually optimize only one of the metrics above. Since our idea of online switch policy has not been fully exploited, further areas of improvement can be investigated. Possible future work could include the extension of our policy to include more general switching conditions, for example using information not related to the progress of exploration.

This work has the following structure. Chapter 2 gives a detailed overview of the state of the art in terms of exploration, coordination, and communication strategies, along with a presentation of indoor semantic mapping and generative models to predict unknown parts of the environment. In Chapter 3, we present the problem of online switch of communication strategies from a formal stance and define the goal of our research. Chapter 4 presents our solution to the problem in terms of an online switch policy between the Multi-hop strategy and the Rendezvous one and discusses the choice of a suitable estimator for modeling the area of the environment. In Chapter 5, the software architecture is presented. In particular we give a short overview of the MRESim software for multirobot simulation, before moving to the implementation details of our policy. Chapter 6 presents the experiments made to test our policy and compares the results in different settings. Chapter 7 summarizes the purpose and the outcomes of this thesis. Some suggestions for future work are also proposed.

Chapter 2

State of the art

In this chapter we present the state of the art in terms of exploration, coordination, and communication strategies for multirobot systems, focusing specifically on the issue of communication strategies. Then, we give an overview of semantic mapping models and how generative models can provide predictions about unexplored parts of environments.

2.1 Exploration, coordination, and communication in multirobot systems

Multirobot systems are systems where a group of robots coordinates themselves to achieve a common goal. One of the most important field in which multirobot systems are employed is the exploration of an unknown environment. Exploration of initially unknown environments is an online task in which autonomous mobile robots coordinate themselves in order to efficiently discover free spaces and obstacles. Often, they are also required to communicate their collected data to a *Base Station (BS)*, which is in a fixed position of the environment (possibly the starting one), where, for example, a human operator can keep track of the progress that are made. In order to achieve these results the system should have:

- An **exploration strategy**, which is used by robots to decide where they should move. The main goal is to move to the *frontiers* of the environment (i.e., the boundary between mapped and unknown space) in order to complete the exploration of the environment minimizing some kind of measures like the time required, as it is shown in [35].

- A **coordination strategy**, with which the group of robots decide where every member should be placed to accomplish the exploration mission. This form of coordination is often developed assuming the possibility of communicating without limit. However in real world settings, this chance is not always given.
- A **communication strategy** that deals with the problem to determine how the robots should exchange information. This part is really predominant in real world application, because the robots are not always given the possibility to communicate with another due to limitations on the communication systems or peculiarity of the environment.

It is important to remark that exploration, coordination, and communication strategies in multirobot systems are not separate entities, but, on the contrary, they are tightly bound. An exploration strategy relies on the presence of a coordination strategy, which works on the assumption of a specific communication strategy employed by the system.

From an abstract stance, the exploration problem can be represented as it is shown in [6]. Assume to have a bi-dimensional environment E and assume time is discretized in steps. A set of n robots, $R = \{r_1, r_2, .., .r_n\}$ is able to move in the E. For any time t, call $p_i^t \subset E$ is the portion of the environment know by robot i. The map, representing the portion of the environment perceived by robots at time $\bar{t}$, is given by:

$$M^{\bar{t}} = \bigcup_{i=0...n} \bigcup_{t=0...\bar{t}} p_i^t \tag{2.1}$$

The exploration is complete when there is a time $\bar{t}$ where $M^{\bar{t}} = E$. The problem of an exploration strategy in multirobot systems is to find, at any time t or after a given temporal deadline, which frontiers (which are the borders between $M^{\bar{t}}$ and E) a robot should reach next, while optimizing a performance measure (such as time required and distance traveled to complete exploration). The decision on which frontiers to choose and where each robot goes form the exploration and coordination strategies and they can be considered the "intelligence" of the system.

This aspect of the exploration and coordination strategies in multirobot systems has been widely studied in literature and it is possible to find two main approaches. The first approach is based on a market principle, in which robots place bids on subtasks of the exploration effort (which are the frontiers of the known part of the environment) as in [29], [10]. These bids

are typically based on values such as expected information gain and travel cost to a particular location in the environment, and the frontiers may be assigned in a distributed fashion among team members, or by a central agent. When strength of communication is factored into the bids, robots avoid areas outside of communication range.

Another common strategy is to use utilities of the frontiers, as it is explained in [35], which can easily be extended for use by multiple robots, like it is shown in [7]. Similar to bids described above, utilities of individual frontiers may include a factor related to likelihood of communication success, so robots are less likely to explore areas that take them out of the team communication range.

Having this idea in mind, we can now apply this system to the case of a multirobot exploration with limited communication as in [24]. The authors present different solutions to solve the problem of exploring an unknown environment under the constraint of a limited wireless networking. For instance, they illustrate a strategy in which the robots move freely in the environment as a pack while maintaining communication with each other.

Different solution can be presented to solve problem of exploration under the assumption of limited communication, but all of them bring same consequences to the whole system. First of all, two robots, at a given time step t, can have different knowledge of the environment, so that there should be a map-merging algorithm that works when the two reconnects. Moreover, this can lead to degradation of performances if the coordination mechanism is not carefully implemented.

Secondly, robots can predict, using a communication model, when and where they are able to communicate with other teammates. However, this prediction could turn out to be wrong and this can have a significant impact on the performance of the system itself. In order to reduce this degradation a conservative assumption over the capabilities of the robots to communicate can be made, as it is shown in [2].

Third, in real world settings, additional constraints can be imposed over the communication of the system. For example, a human operator could be interested to receive periodically a video stream from robots cameras. This scenario can lead to systems in which all the robots need to build a chain of locally communicating teammates to relay the video stream from the point

of interest, as in [18].

To investigate this problem further on, we start by illustrating the communication models addressed in the literature and then moving to explain the main communication strategies.

2.1.1 Communication models

A typical mobile robot used in exploration tasks is usually able to exchange information with its teammates and possibly with a Base Station (BS), over a radio channel, such as WiFi, whose quality degrades with the distance and the presence of obstacles. Given this fact, the *communication model* refers to the prior knowledge the robot has on the communication capabilities, which it exploits during the decision on where to go next. This decision can be critical because, for instance, a robot can predict to be able to communicate with another robot only basing on the fact that they are mutually visible and within some range. This communication model can lead to a robust multi-hop chains in case we need a stable video stream between a location of interest and the BS. As it is shown in [6], the communication model can be categorized in:

- **None**: the robots do not make any assumption on whether they are able or not to communicate between two given positions.

- **Line-of-sight (LoS)**: a robot can communicate with one of its teammates if the line connecting their positions is entirely contained in the free space of E, as it is shown in [2]. Usually the line has also a maximum length d.

- **Circle**: a robot can communicate with one of its teammates within a fixed maximum distance d, whether or not the presence of obstacles, as it is shown in [28].

- **Signal**: a robot r_i can communicate with a robot r_j with a probability that depends on the estimated signal power between the position of r_i and the position of r_j. This model takes into account the degradation of the signal. for example derived from the encounter of obstacles. This model is further investigated in [34].

2.1.2 Communication strategies

The other dimension related to the communication is the connection requirements to which a system should comply to accomplish the exploration goal. This could be divided in three main categories:

- **None**: robots are not required to communicate.
- **Rendezvous (RV)**: robots must regain connection with teammates according to a particular policy triggered by some kind of events, such as the discovery of new information about the environment, or simply striking a particular time.
- **Multi-hop (MH)**: each robot must maintain connection with its teammates (as well as the BS), either in a direct connection or in a multi-hop fashion.

We can note that the most interesting strategies are the Rendezvous and the Multi-hop ones, because they add increasing levels of constraints on how the exploration strategy can choose the prominent frontiers to visit.

It is important to note that, to best of our knowledge, all the multirobot systems proposed in the literature employ only one of the above communication strategies at a time. For example, a multirobot system can explore an environment using a Multi-hop strategy from the beginning to the end of the mission or otherwise it can use a Rendezvous strategy. This, as said before, can have an impact on how the exploration goes on and how the system performs. There are some systems where both strategies can be used, as in [1], but the switch is done off-line, so that, during an exploration run,the system just uses one of them.

2.2 Multi-hop strategies

Multi-hop (MH) strategies are communication strategies that comply with the constraint that all the robots and the base station must be always connected one another. This type of constraint can be seen in two different ways: either it can be a continuous connectivity constraint or a recurrent connectivity one.

The constraint of *continuous connectivity* means that all the robots must be connected with each other throughout the whole exploration of the environment. Moreover, they must maintain connection with the Base Station

for the same time. This constraint is ideal for situation in which an operator at the BS wants to have a video stream of a point of interest or simply keep track of the exploration in real-time.

A specific implementation of this strategy is explained in [18] and it is generally known as *Leader-Follower (LF)*. The authors address the problem of maintaining a robust high-bandwidth radio frequency communication link between a mobile robot and its BS. The solution uses a number of autonomous mobile relay nodes to form an ad-hoc network. The followers follow the leader and automatically stop where needed to maintain connectivity between the lead robot and the BS.

A similar solution is addressed in [1], where the robots try to cope with some teammate failures and form a robust network where the failure of a relay does not influence the connectivity constraints. The robots form a robust ad-hoc network where there is at least two possible paths to communicate with any robot. In this case, if one of the relays suffers a failure the BS remains connected with the leader.
Another important achievement of this work is the ability to have multiple leaders at the same time. The system deploys at any given time the maximum possible leaders according to the constraint of robustness and continuous connectivity.

The second constraint that falls in the MH strategy is the *recurrent connectivity constraint*. This constraint can ensure situational awareness of the robots without excessively constrain the exploration task like the continuous connectivity. In this method, the robots are required to be in communication any time new information is gathered. Under such a scheme, the constraint can be thought in an online fashion, allowing the robots to be disconnected for arbitrary long periods of time, if during that time no new information about the environment is discovered and provided that they are able to communicate with the BS as soon as new information is collected. However, it can be useful to note that time frame when the robots are disconnected is usually small and a disconnection can happen only when a robot has to move from a location already fully explored to a new unexplored one.

In the system explained in [5], the authors build a multirobot system under the recurrent connectivity constraints. The solution proposed constructs a graph keeping track of the frontiers already explored and of the current ones and it deploys the robots in a way that they are always connected to

the BS and at the same time they explore the most promising frontiers. This deployment is computed using an approximate solution of the Steiner tree problem(i.e, given a subset of vertices, find a minimum-cost subtree that connects them) and then assigning the robots to the vertices minimizing the total distance traveled.
An interesting feature of this solution is the ability to compute the deployment in an asynchronous way, meaning that new plans can be submitted to arbitrary groups of robots as soon as they become ready. This feature can speed up considerably the performance of the whole multirobot system in terms of time taken for exploration.

2.3 Rendezvous strategies

Rendezvous (RV) strategies are communication strategies that work under some kind of event-based connectivity constraint. This type of constraint means that the robots are free to move in the environment, accordingly to their exploration and coordination strategies, but they must regain connection with a group of their teammates or the BS as soon as a particular condition is met which is triggered by some kind of events. The strike of this task can be triggered essentially in two ways, either by the substantial gain of new information or on a periodic time basis. The term *rendezvous* itself has been proposed in [26], with the meaning of a meeting point for a group of robot. Here it is used with a wider approach to denote all the strategies that deal with the aforementioned constraints.

RV strategies can be divided into different groups based on the agreements the robots have on the way they must reconnect. It is important to consider that most of the RV strategies already implemented are built on the following assumption. It is not needed that the totality of the robots has to reconnect when the event is triggered, but only the agents that have reached an agreement on that particular event have to regain connection. Given this fact, the two main groups in which we can divide the Rendezvous strategies are the implicit and explicit ones.

2.3.1 Explicit RV strategies

An explicit RV strategy is an agreement between a group of robots on where to meet when an event that triggers the reconnection occurs. This type of

strategy can work on the top of different types of events, such as periodic reconnection events or the gain of new information. A good example of a fairly simple RV strategy is [12], where the authors change the continuous connectivity constraint in favor of a more relaxed version which implies a periodic reconnection by the group.
The idea lying beyond the approach is the following: at fixed time intervals all the robots must reconnect themselves and possibly with the BS in order to share new information they gained exploring the environment on their own. Then they agree on where everyone should continue their exploration and decide when and where the next meeting should happen.

Explicit RV strategy can be seen also in a different way. The authors of [8,9] propose a strategy, called *Role-Based Exploration (RB)*, in which every robot of the group is assigned with a specific role, either it is an explorer or a relay. The task of an explorer is to explore the farthest reaches of the known environment, while the task of a relay is to ferry information back and forth between explorers and the BS, by meeting the explorer periodically at a rendezvous point and returning to the BS. The agreement between robots of different roles comes in the form of when and where to meet. The relays and the explorers compute a point, called rendezvous point, where they are able to meet and then set a period of time in which the explorers try to gain new knowledge on the environment and the relays go back to the BS to communicate the newest information.

This basic structure can be improved adding a more complex hierarchy in terms of explorers and relays. For example, if the rendezvous point is very far from the BS, the system can build a chain of relays to exchange information faster. This approach has been also shown in [1], which builds a dynamic chain of relays to support the exploration of a group of explorers. In RB exploration a crucial point is the selection of the rendezvous point, which can lead to a degradation of performance if not carefully chosen. In [13], the authors introduce a novel way of computing them that can be also extended to situations where the environment changes dynamically, while the exploration takes place, for instance due to the collapse of a wall in an hazardous setting.

2.3.2 Implicit RV strategies

On the other side, there are the implicit Rendezvous strategies. This type of communication strategies differ from the explicit ones because the robots do not agree by themselves on a location or a time to meet, but on the contrary they agree only with the BS to fall back to report new information as soon as an event that triggers the reconnection occurs. In this type of strategy, the event can be either a periodic time reconnection event or a more sophisticated one, like the discovery of information about the environment.

The strategy proposed in [30] falls in the latter category. The authors propose a system where all the robots are considered explorers, so that no hierarchy is given a priori. Any robot agrees with the BS, at the beginning of the exploration, to go back when the *information gain* related to their knowledge of the map is greater than a given threshold. The robots are given the possibility to communicate and coordinate in a completely distributed way, so that, if two robots meet, they can merge their maps and update their own knowledge of the environment.

From a formal stance, the strategy works this way. Before the start of the exploration the user sets a target information ratio $targetInfoRatio \in (0; 1]$, which gets propagated to all agents in the team. For each agent i, let $infBase_i$ be the information i believes the BS to have in a given time. This value is obtained from communicating with the BS itself or with other agents that have communicated more recently than agent i. Let $infNew_i$ be the information about the environment that i knows excluding $infBase_i$ and excluding information that the robot i has give to other agent to relay to the BS.

During the exploration task, each robot can be in two stance: either it is exploring the environment or it is returning to the BS to relay the information. A robot i only decides to fall back to the BS if

$$\frac{|infBase_i|}{|infBase_i| + |infNew_i|} < targetInfoRatio \tag{2.2}$$

where $|infBase_i|$ and $|infNew_i|$ are the utilities of $infBase_i$ and $infNew_i$ accordingly.

An interesting behavior observed in this system is the following. For higher values of *target information ratio* and high number of agents, at the start of exploration all the robots go off to explore new frontiers and collect

information. However, as the exploration effort goes deeper into the environment, a number of agents end up acting as dedicated relays, simply going back and forth between the base station and the other exploring robots. At late stages of exploration the agents tend to form a chain of relays, where only few robots are explorers, as if they were in an explicit RV strategy, but in this case there is no explicit agreement on where or when the robots should meet.

This strategy behaves as follows. A number of robots meet while returning to the BS to deliver their information. While they are in communication range, they exchange information and the robot nearest to the BS assumes the responsibility of delivering back the combined new information back. After that, the "relay" agent moves towards the area with the most promising frontiers, but since it has the same knowledge of the agents it previously met, it will likely go to the same frontiers the other robots are headed, meeting them, or another robot, on its way.

In terms of performance, as the authors note in [30], the system is very scalable. Even with using a high value of $targetInfoRatio$ (which means that the robots have to return more often to the BS), it is possible to note that the whole system, in most of the settings, performs better or at least it has comparable performance with an explicit RV strategy, where the robots need to meet on a periodic time basis. The major drawback, however, is in terms of assurance of a periodic update at the BS, due to the fact that the system does not take into account any time constraint while computing when a robot should go back to relay the information. However, using higher values of the parameter constrains the robots to return more frequently to the BS and so having a relatively small time interval between two consecutive updates at the BS.

2.4 Semantic mapping and generative models

In robotics,maps are broadly used to represent the environment in which a mobile robot or, in general, an intelligent agent has to operate. Such maps can represent different aspects of the environment as, for example, the metrical description of a level of a building or the information about the structure of the environment in terms of rooms and corridors. According to [22], a map can belong to one (or more) of these types:

- **Metric map**: it describes the environment showing which areas are free and which ones are occupied (as an example, by the presence of an obstacle, such as a wall). There are numerous ways to represent a metric map: the most used is the grid map, in which the environment is divided in a set of regular cells, which contain the probability to encounter an obstacle.
- **Navigation map**: it describes the environment in the form of navigation goals by a robot, which can be specific locations in the environment. It is represented as a graph built as the robot explores the environment, where the nodes represent free space and edges are paths from a free space to another. A new free space navigation node is placed in the map whenever the robot has traveled a certain distance from the closest existing node. The final graph serves for planning and autonomous navigation in the already visited part of the environment.
- **Topological map**: it describes the environment in terms of a graph consisting of nodes and links, which represent rooms in the environment and their connectivity, respectively. The structure of the topological graph is built based on the assumption that the transition between two rooms happens through a door. This allows to reason on a human-like qualitative segmentation of an indoor space into distinct regions.

It is useful to note that all these maps, beside the metric one, are built on the top of the previous ones. For instance, the navigation map is linked with the metric one, because each node of the graph is assigned a position in metric coordinates, while the topological map can be seen as a more abstract representation of the navigation graph, where the nodes are grouped by their position in the environment.

2.4.1 Semantic mapping of indoor environment

Semantic mapping of an indoor environment refers to the task of building representations of this environment that associate spatial concepts with spatial entities. In particular, it is possible to associate labels (for instance, they represent the function of a room, such as kitchen or bedroom) to a node in the topological map. Usually this type of maps are paired with the others to leverage their information in various fields. For instance, a robot can use knowledge obtained from the metric map (such as the frontiers) combined with the semantic information about the environment in order to improve

path planning or exploring first the important rooms of the building.

A well known way to build semantic maps is to use information on sensors installed on a mobile robot or, possibly, on a multirobot system. An example of this approach is given in [17]. The author presents different approaches to enable a mobile robot to categorize places in indoor environment. All of them are implementation of supervised learning techniques, such as *AdaBoost*. The objective is to give the robots the possibility to represent the environment in a similar way humans do.

The process of building semantic maps, however, does not only rely on information extracted from sensors. In [11], the authors propose a system, where a metric, topological and semantic representation of the environment is coupled with natural language information derived from sensors like microphones. The algorithm is able to extract semantic proprieties of human-centered concepts combining the natural language descriptions with images and laser-based classifications. This leads to an improvement of the accuracy of the achieved spatial-semantic representation of the environment.

Semantic mapping can be used as one of the inputs of an exploration and coordination strategy. In [32], the authors consider this problem in the setting of a multirobot system. To better distribute the robots over the environment and to avoid redundant exploration of the same places, they take into account the type of place a potential target is located in. This approach can lead to better performance compared to a traditional coordination strategy.

A particular type of the exploration problem is the *search and rescue* setting, where robots need to explore an hazardous environment finding possible survivors. Semantic mapping can be useful there, as it is shown in [23]. The authors leverage the information given a priori by semantic mapping to to push robots to explore relevant areas of initially unknown environments. The system also proposes to speed up the exploration of specific areas by using semantic information both to select locations to visit and to determine the number of robots to allocate to those locations.

Most of the methods used to perform semantic mapping of an environment use information extracted from sensors and Machine Learning techniques, such as supervised learning, to extract useful features to classify portions of the environment. An example is presented in [25], where the au-

thors use a boosting technique and train a classifier with features extracted from vision and laser range data. This approach turns out to be efficient in labeling parts of the environment already visited by the robots, but it fails in predicting the label and the topological structure of the environment. This is due to the fact that all these methods assign labels on the basis of sensor readings acquired in the current environment and of models trained on sensory data acquired in previous runs of the robot.

2.4.2 Generative models

As already said, while semantic mapping models perform well in standard settings, one of their main drawbacks is their little ability to reason on the unvisited part of the environment. To overcome this limitation, some extensions to the semantic mapping framework have been proposed. For instance, in [21], the authors present a probabilistic semantic mapping framework that applies semantic labels to different kind of environment features. The map is represented as a probabilistic chain graph model and is adapted at runtime according to the data perceived by the sensors. The chain graph is able to predict the existence of a feature in the unknown environment and can extend the semantic map accordingly. For instance the method is able to predict the presence of the presence of a room of certain category in the unexplored space. In order to do so, the method hypothesizes possible rooms configuration in the unexplored part of the environment and then return a probability that a room of a given category is present summing the probabilities in each hypothesized configuration.

Another approach is reported in [11], where the system integrates metric, topological, and semantic representations with information derived from natural language descriptors obtained from human users. This is done using a factor graph formulation of the semantic properties and inferring these properties by combining natural language descriptions and image- and laser-based scene classification. The result is a method that can infer the existence and location of room given a natural language sentence.

A totally different approach is the one that belongs to the category of generative models. A *generative model* is a mathematical instrument which models an unknown distribution and can be used to sample data from that distribution, starting from given parameters. This category of models is completely different from the previous ones, because it does not only rely

on any sensor data as a direct source of knowledge for semantic labeling. Instead, it operates at a higher level of abstraction reasoning on rooms contained in a knowledge base, that can be obtained without the use of robots.Subsequently, information from sensors data can be integrated to make the prediction more accurate.

This type of integration is what we can find in [3], in which the authors consider a knowledge base containing more than $38,000$ rooms representing university campuses. Each floor of the building is represented as a graph, where nodes are rooms labeled according to their function and edges refer to a direct path between rooms. The most frequent patterns of rooms are then extracted using an algorithm for finding the most frequent subgraphs. Using this information, and given a partial map of a known area, the proposed method predicts both the topology and the labels of unvisited rooms by identifying the most common subgraphs that partially overlap the partial map.

We can find a similar approach in [15], where the authors propose a system based on a generative model that is able to represent the topological structures and the semantic labeling schemas of buildings, while it is also able to predict the structure and the schema for unexplored portions of these environments. The environments are represented as undirected graphs where nodes are rooms and edges the connections between them. Given an initial knowledge base of graphs and relying on a spectral analysis of these, the system segments each graph for finding significant subgraphs and clusters them according to their similarity. The prediction on the unvisited part of the environment is eventually generated by sampling subgraphs from clusters and connecting them.

The method proposed in [14] is a direct application of [15]. The authors proposed a generative model that follows the general pattern of Constructive Machine Learning (CML), where the ultimate goal of learning is not to find good models of the data but instead to find one or more particular instances of the domain which are likely to exhibit desired properties, while selectively sampling from an infinite or exponentially large instance space. A building is represented as an undirected graph which shares the same properties of the previous one. Then the authors implement the method described in [15], distinctively focusing on building types, namely on specific classes of building that share the same structures, as it is presented [16]. Though this method needs better results, it advocates the potential of using high-level semantic

knowledge to predict the structure of unknown parts of indoor buildings in order to improve exploration and search.

Chapter 3

Problem formulation

In this chapter we present a formal definition of the problem addressed in this work. We start by defining the area of research and then we list the assumptions underlying our work and set the thesis goals.

3.1 Multirobot exploration

As we already stated in Chapter 2, one of the most important fields in which multirobot systems are employed is the exploration task in an unknown environment. Multirobot systems need exploration, coordination, and communication strategies in order to be able to manage the exploration task assigned. The first two categories of strategy (namely, exploration and coordination) can have different practical implementations, but they essentially leverage on information about the environment in order to be as fast as possible in fully discovering the environment, while trying to travel the less distance as possible in order to save energy.

This problem requires some kind of trade-off in order to find a suitable solution. It can be easily imagined that not always the fastest strategies can lead to the least distance traveled by the whole system. The difficulty in finding a good solution increases if we consider that the whole system has to comply with a communication model and a consequent communication strategy.

In the literature, different solutions have been proposed to tackle the problem of dealing with a communication constraint while exploring the environment. This constraint often takes into account that the robots should

maintain a connection with a Base Station to keep it informed. This connection can be in a form of continuous or recurrent connection or it can be an event-based connection. The solutions adopted can fall in a Multi-hop (MH) strategy, if they deal with the continuous constraint, or in a Rendezvous (RV) strategy, if they deal with an event-based constraint, we have already seen in Chapter 2. It is important to remark that, to the best of our knowledge, no system is able to decide, at run time, which strategy needs to be adopted in order to achieve the best performance.

3.2 The problem and the goal

The work we propose tries to cope with the problem of building a multirobot system which is able to explore an unknown environment using different communication strategies. We aim to identify under which conditions one strategies should be preferred over another. Before explaining in a more rigorous and detailed way the formulation of the problem, we are going to present some important assumptions we have made in this work.

3.2.1 Assumptions

Robots The first important set of assumptions we have to make is about the robots. In this work we consider to have a multirobot system in which a team $R = \{r_1, r_2, ..., r_n\}$ of mobile robots are exploring an initially unknown environment. Moreover we have another agent, which plays the role of a BS, at the starting position of the exploration. The BS is incapable of moving and it needs to receive updates about the mission status and the exploration task, so it must be able to communicate with the robots.

The robots are able to move around in the free space of the environment and they are equipped with sensors, such as laser scanners and cameras, which are the input of the SLAM (simultaneous localisation and mapping) module. On the other side, the BS does not have any sensor and it relies only on the robots to perceive the environment. We assume the SLAM module to provide reasonably accurate localisation. This can be considered a realistic assumption, because recent approaches, such as those purely basing on scan-matching, as it is explained in [20], or on particle filters (like in [33]), are considered robust when dealing with noisy data. This assumption gives us the possibility to focus our work specifically on the decision process, instead

of dealing with noisy data that require filtering.

Communication A second set of assumptions can be made about the communication between robots (including the BS). All the robots ()and consequently the BS) are equipped with RF transceivers (e.g., WiFi). These transceivers are typically assumed to have an unlimited bandwidth channel in which the robots can exchange information. In the literature, it has been shown that this assumption can hold even in real setting. For instance, in [5], the authors show that this assumption does not affect the performance of a real multirobot system.

The transceivers have a maximum communication radius d which can vary according to the environment. We employ a communication model to infer the possibilities of two robots to communicate with each other. We decide to adopt the Signal strength model, presented in Section 2.1.1, in which two robots can communicate based on the signal strength power between their positions. The communication strength we adopt is defined in [4] and works as follows:

$$S = P_{d0} - 10 \times N \times \log_{10} \frac{d_m}{d_0} \times \min(nW, C) \times WAF, \tag{3.1}$$

where P_{d0} is the signal strength at the reference distance d_0, N is the rate of the path loss, d_m is the distance, nW is the number of obstructing wall, WAF is the wall attenuation factor, and C is the maximum number of wall where the attenuation factor needs to be considered.

We use this model because it takes into account the degradation derived by the encounter of obstacles and so it models a real setting in a better way. Another reason to choose this communication model is the possibility to further constrain it in special cases, for example, using a shorter distance Line-of-Sight. We assume, moreover, the communication to be perfect. This means that if the model guarantees that two agents can communicate, they are actually able to do it and there is no loss of information over the channel, meaning that all the information sent is always delivered.

We want to add another important assumption about the communication. We assume that the robots and the BS are equipped with a second communication channel. This special channel has a very limited bandwidth, but a very large range of communication and it is able to cover the whole environment. This communication channel cannot be used to send normal

information about the environment, such as the partial maps, but the BS can use it to alert all the agents of particularly critical situations that have just happened.

Environment A last set of assumptions has to be made over the environment. In this work, we are going to focus our studies only on a subset of all the possible environments. The environments we choose are indoor environments. These environments can have different shapes and different layouts, but they all share the feature of being formed by spaces that can be essentially mapped in rooms or corridors. Moreover, we assume that these environments are static during the exploration task, meaning that they are not changing over the progress of the exploration. Lastly, we assume these environments to two-dimensional and formed of a single floor. However, extensions of the following method to three-dimensional environments is straightforward.

3.2.2 Online switch of communication strategies

As already mentioned before, the aim of this work is to build a system that is able to work under different communication strategies during the exploration of an unknown environment. We call this feature online switch. An *online switch* is a particular policy that enables a multirobot system to work under different communication strategies. An important remark of this policy is the ability to switch the communication scheme during the exploration task of the robots, without having to stop the system and setup again a new connection. This behavior places our approach on an orthogonal plane with respect to the literature characterization of communication strategies. The strategy produced by our policy, as a matter of fact, cannot be considered neither a pure Rendezvous strategy nor a pure Multi-hop one. On the contrary, it originally works using strategies from both of the categories in the same exploration run.

A good choice for an online switch policy is to change between a MH strategy and a RV one. For instance, consider a system that needs to minimize for efficiency reason the time not in communication with the BS and it deploys the robots using a MH strategy, so it can have a reliable stable channel to send the information. During the exploration run, the system wants to change communication scheme to be able to minimize the time taken to explore the environment. Since this metric is conflicting with the

previous one, the system decides to find a suitable trade off. The robots are no longer required to have a continuous connection with the BS. A MH strategy is not able to relax this constraint to free the robots to explore the environment. Our policy, on the other side, can switch the communication scheme to a RV strategy in order to have only periodic updates at the BS. Doing this, the robots have more freedom to explore the environment without having to maintain connection with the BS. If, at the end of the exploration, the system needs to form again a multi-hop chain, the robots will regroup and switch back to a MH strategy. The signal to inform the robots to regroup can be broadcast over the second communication channel, in order to shorten the time needed to regroup.

The system decides to switch between the two different communication strategies using a *policy*, which analyzes the state of the system Env, as well as the mission status M, and returns which communication strategy the system must employ. We define the set of all the N possible communication strategies, $CommStrategies = \{cs_1, cs_2, ..., cs_n\}$. At any time step t the system evaluates the information it receives from the environment, the status of the mission, and the previous communication strategies to decide the communication strategy it needs to employ, cs_t. The policy $switch$ is defined as it follows,

$$cs_t = switch(Env, M, cs_{t-1}), cs_t \in CommStrategies, \tag{3.2}$$

where $cs_{t-1} \in CommStrategies$, is the previous communication strategy. $cs_t = cs_{t-1}$ means that the system is not required to change the communication and it keeps using the previous one, while $cs_t \neq cs_{t-1}$ means that the system has to actually switch from a communication strategy to another.

For instance, if we recall the aforementioned example of the switch from a MH strategy to a RV one, the set of the possible communication strategies is formed by $CommStrategies = \{MH, RV\}$. The switch between the MH strategy and the RV strategy happens when $switch(Env, M, MH) = RV$ and conversely for the other switch.

3.2.3 Goals

The primary aim of a switching communication strategy is to give the system the capability to adapt its communication strategy in order to react to particular features that can arise during the exploration of an unknown

environment. For instance, we can consider a particular exploration setting, such as search and rescue. In this setting, it can be useful to have a video streaming of an interesting target, while the robots are free to move in the environment. The strategy produced from our policy has the ability to deploy an *ad hoc* MH strategy for the only purpose of the video stream, while for the rest of the exploration the robots can move using a RV strategy.

Another important aim is the following. In the literature, it has been shown that based on which communication strategy is used, the performance of the system can be different, as we can see explained in [1]. The classical performance metrics, whose are the time taken to fully explore an environment, the distance traveled by the robots, and the time not in communication with the BS, are conflicting objectives to minimize, so it does not exist a strategy that can minimize all of the three above. For example, the MH strategies are meant to minimize the time not in communication, but doing this the time taken to fully explore the environment is usually greater than the time taken using a RV strategy, which works in the opposite way. Our combined strategy, because of the heterogeneous nature of the communication scheme, presents a trade-off that will be compared with state-of-the-art strategies according to the above metrics.

Chapter 4

Proposed solution

In this chapter we present the solution elaborated to the problem formulated in Chapter 3. We present a logical view of the solution implemented given an instantiation of the original problem.

4.1 Switch strategy between MH and RV

In Chapter 3, we have given a formal definition of what an online switch policy for communication strategies is and we have defined the goal of our research along with the main modules needed to accomplish it. Here we present a specific implementation of the policy for the considered switch strategy, in the form of a switch between communication strategies of different categories. Recalling what said in Chapter 2, the communication strategies can be divided in two main categories, which are Multi-hop strategies and Rendezvous strategies. Our choice of implementation is to choose two communication strategies, one for each category, and instantiate a switch communication strategy following the formal definition given. We start with discussing the selection of the Multi-hop and Rendezvous strategies.

4.1.1 Selection of Rendezvous strategy

In Chapter 2, we have given a broad presentation of what, in the literature, we can consider a Rendezvous (RV) strategy. A RV communication strategy works under the periodic connectivity constraint, which gives the robots the capability to explore the environment farther than their maximum communication range while being able to regain connection with the BS to inform

it of the exploration status on a periodic basis or when a particular event occurs.

In this work, we rely on a specific implicit RV strategy, which is the one proposed in [30]. This strategy does not instantiate a hierarchy in the team of the robots, considering all of them as explorers. The robots agree with the BS to go back to communicate the information gathered during exploration only when the amount of collected information is greater than a given threshold decided at the start of the exploration. Following this strategy, an emergent behavior appears. Some robots, even if they were all initially considered explorers, begin to act as dedicated relays going back and forth between the BS and the prominent frontiers where, during this travel, they meet the other teammates. Furthermore, they often behave as a chain of relays at the later stages of exploration.

In [30], the authors compare the performance obtained with this strategy against that obtained with an explicit RV strategy, which is called Role-Based exploration (RB), that can be found in [9]. In RB exploration, every agent of the team has a defined role (i.e., it is a relay or an explorer). The results show that the implicit RV strategy is faster to fully explore the environment than the explicit one. This is thanks to the fact that it does not designate a number of agents to be used only as relays throughout the exploration run, which results in a more efficient use of resources. Moreover, the comparison of the total knowledge at the BS, when the robots reach the complete exploration, yields similar results. This can lead to time not in communication between the agents and the BS similar in the two strategies. Based on these observations, we choose the strategy explained in [30], based on the fact that it is able to explore the environment faster without sacrificing too much the connectivity.

4.1.2 Selection of the Multi-hop strategy

The second strategy we choose belongs to the category of Multi-hop (MH) strategies. Recalling the literature, a communication strategy is considered to be a MH one when it deals with a continuous connectivity constraint or a recurrent connectivity one. While the former one constrains the robots to be always connected to the BS, the latter one requires the whole team to be in communication with the BS only when an agent is at a frontier, giving the possibilities to arbitrarily long disconnections while the robots are moving

in already explored parts of the environment.

As a candidate of this category of strategies, we choose the one presented in [5], which works as follows. The authors propose a method to deploy a connected chain of robots while they explore an unknown environment. An important achievement, stated by the authors, is the ability of the system to compute the deployment of the robots in an asynchronous manner, as soon as a set of robots is "ready" (i.e., it has accomplished the goal given by the BS).

We choose this connectivity constraint and this particular strategy for a number of reasons. The choice of a recurrent connectivity constraint is made because, even if it does not guarantee a connection at any time step, it allows a good situational awareness at the BS. As a matter of fact, the disconnections are always temporary and for short periods of time. In [5], the authors compare their method with the one presented in [30]. Surprisingly, their performance is similar if the system redeploys the team as soon as one robot is ready. Based on these observations we choose the strategy explained above as our MH strategy.

During the development of the MH strategy, we apply a further constraint on the MH part of the communication. As stated in Chapter 3, we use the Signal strength model to represent the presence of a communication channel between two agents. In this phase, we constrain the communication model of the MH strategy to be as much conservative as possible while doing its deployment. We choose to adopt a Line-of-Sight (LoS) communication model for the deployment. This means that two robots can communicate only when they can see each other within a maximum distance d. The deployment uses this model to compute a conservative placement, underestimating the communication capabilities of the whole team. We make this decision in order to avoid possible situations in which the model states that the communication is possible, while actually it is not.

4.1.3 Connection issues during the switch

The switch between the two strategies is not a straightforward process. Both the switches, i.e., from RV to MH and *vice versa*, have some technical difficulties to overcome. Most of them are due to the completely different nature of the communication scheme they employ and to the need to exchange of

information to coordinate the switch.

The easiest switch to perform is the one when the system changes from a MH strategy to RV one. The reason is that we can simply relax the connectivity constraint. Moreover, at the time of the switch, the robots are all connected to the BS, therefore all the robots can immediately be aware of the fact that the switching is taking place. The only technical problem is due to a peculiarity of the MH strategy. In order to compute the deployment, the strategy keeps a list of the frontiers visited and uses them as possible placements for the robots. The RV strategy needs to be extended to keep track of the frontiers visited by each robot and to communicate them to the BS when they are in communication.

The opposite switch (i.e., from the RV strategy to the MH strategy), needs further precautions. We assume to have an up and running RV strategy, in which every robot keeps track of the frontiers it has already explored and, any time it communicates with the BS, delivers the list of the frontiers visited. When the policy at the BS decides to switch to the MH strategy, the system moves to a *wait state*. In this state, the BS communicates to all the robots connected to employ a MH strategy. Since not all the robots can possibly be connected at that time, the ones that are in range stop their tasks and wait until all the team regain connection with the BS. This happens because some robots at that time are outside the communication range of the others and the first time they regain connectivity and become aware of the change in the communication strategy is when they go back to the BS to deliver information.

The aforementioned behavior happens due to the fact that the system moves from a relaxed version of the connectivity constraint to a more tightened one. If we assume to have only a single communication channel with a limited range, the only possibility for completing the switch from RV to MH is to wait until the last robot has eventually regained connection with the BS. However, if the system can employ a second communication channel, which meets the requirements explained in Chapter 3 (i.e., sufficient range to cover the whole environment, but limited bandwidth), the BS can leverage this channel to alert in a synchronous way all the robots of the team. This works as it follows. The BS, once it knows it has to switch to the MH strategy, broadcasts an alert message over the second communication channel. All the robots receive the message, stop their current task, and move back to the BS. As soon as the last robot is connected, the BS replans

the deployment of the team following the MH strategy.

4.1.4 Policy selection

The last part left to discuss about our solution, is the development of a suitable policy. Recalling what said in Chapter 3, a policy is a function which gives as output the communication strategy the robots have to use according to the current state of the mission. In order to have a running implementation of a switch communication strategy, we need a policy which determines what communication strategy has to be employed at any time step.

We define our policy starting from the observation reported in [1]. The author notes that different communication strategies have better performance according to the stage of the exploration. This means, that it does not exist a communication strategy which outperforms all the others in an exploration task.

Based on this, the author of [1] raises an hypothesis for which MH strategies are better suited for the earlier and later stages of the exploration, while RV strategies perform better in the middle of the exploration. This can be intuitively explained as it follows. At the beginning of the exploration, the two strategies have similar performance, because robots starts close to the BS, so we have a high number of explorers and a low number of relays in MH strategies, and high frequency updates in RV strategies. As the exploration goes deeper, MH strategies need to employ more relays to maintain connection with the BS and less robots can actually explore the environment. On the other hand, RV strategies are not required to maintain a connection with the BS; this frees up more robots to move in the environment pursuing their exploration tasks. At final stages of exploration, it usually happens that only small parts of the environment are still unknown and a small number of robots can easily explore them. Relays in the RV strategies are thus required to cover great distances to travel back and forth rendezvous points, while in MH strategies the information is delivered instantaneously to the BS, speeding up the conclusion of process.

We follow this hypothesis and model a policy in which a MH strategy is used at the beginning of the exploration, followed by a RV strategy which switches back to the Mh strategy while the system approaches the later stage

of the exploration. To implement this behavior, we define two parameters, α and β, fixed at the beginning of the exploration. α represents the threshold used to move from the first MH strategy to the RV strategy. β represents the threshold used to switch back from the RV strategy to the second MH strategy. Recalling the notation of Chapter 3, the set of strategies will be $CommStrategies = \{MH, RV\}$. The system decides to employ a different communication strategy following this scheme:

$$switch(Env, M, curr) = \begin{cases} RV, \text{ if } \alpha \leq f_{sw}(Env, M, curr) < \beta \\ \\ MH, \text{ otherwise} \end{cases}, \quad (4.1)$$

where $f_{sw}(Env, M, curr)$ is a function we use to model the status of the exploration and $curr$ is the current communication strategy.

4.2 Modeling the prediction

The last part of our solution focuses on how we can give a suitable value to the switch parameters α and β, and consequently to $f_{sw}(Env, M, curr)$. These two parameters should be related to the progress of the exploration, specifically representing a threshold on the percentage of area already explored (hence, $\alpha, \beta, f_{sw}(Env, M, curr) \in (0; 1]$). In order to this, we must have an a priori knowledge of what is the extension of the environment against which the amount of mapped are is compared. Obviously, the total area the robots have to explore is unknown and, therefore, it is impossible to compute the percentage of the environment already known, based on the real area of the exact environment.

Starting from this assumption, we find a suitable prediction of the total area of the environment, $\widehat{A}$, using different models. From them, we derive, at any time step, an estimate of the percentage of the area already explored, which we use as our $f_{sw}(Env, M, curr)$, as it follows,

$$\widehat{f}_{sw}(Env, M) = \frac{ExplArea}{\widehat{A}}, \quad (4.2)$$

where $ExplArea$ is the amount of area of the environment explored so far. This value is later compared with parameters α and β in order to decide which strategy the system should employ.

The models we present use features of the environment that can be observed without fully exploring it. They leverage a priori knowledge on the type of the environment using semantic mapping and generative models.

4.2.1 Data analysis

In order to find interesting features to use in our prediction models, we focus on a specific type of environments, which is school buildings. Some previous works, presented in [14, 15], has already focused their attention to this particular building type. In particular, the authors present a generative model which is able to make predictions about the unexplored part of the environment, starting from an explored subset of it.

The dataset we use consists of about 50 floors from different schools. For all of them, we have the area of the floor and at least one full exploration run on it. The exploration runs are sampled three times each and for every sample, the method presented in [15] extracts predictions about the unknown part of the environment. We process these data to extract the percentage of the environment known when the sample is made, the prediction of the total area of the environment given we have explored that part of the environment, and the error between the prediction and the real area. Starting from these data, we define three estimators to infer the value of the percentage of the area known.

4.2.2 Bounding box estimator

The first estimator is the *bounding box* estimator. This model does not exploit any knowledge derived from the dataset, but, on the other hand, it leverages information that can be derived directly from the environment. Suppose to have the possibility to receive some aerial pictures of the environment from above. These pictures show the general shape of the buildings, but they do not give any information about what the robots are going to find inside the building.

The bounding box estimator takes as input this information to compute the *minimum bounding rectangle*. This is the rectangular box with the smallest area that completely contains the outer perimeter of the building. We take the area computed in this way as our prediction of the actual area

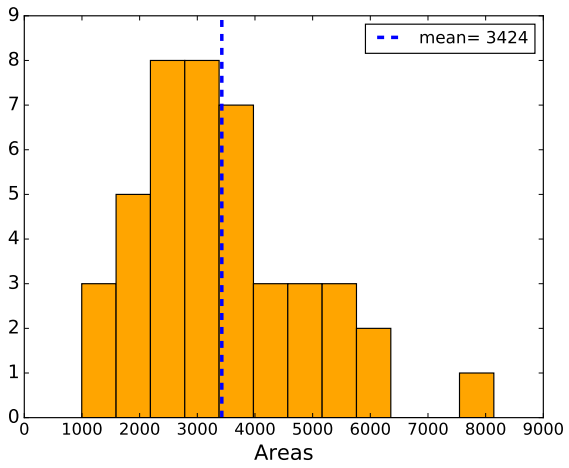


Figure 4.1: Distribution of areas (in m^2) of the environments in the dataset

of the environment and we derive the percentage known from it.

The prediction we extract from this method is always an overestimate of the actual area of the environment, because it simply considered the smallest box fully containing the environment. Hence, the area will be always larger than the actual value, because the estimator does not take into account the shape of the environment. Therefore, this method is expected to yield good results when it deals with environments that have a rectangular-like shape without "holes" (i.e., they do not have any unexplorable part inside the building). In those cases, the bounding box estimator is expected to return a predicted area for the environment which is only a slight overestimate of the actual area.

4.2.3 Mean area estimator

The second estimator is the *mean area estimator*. The method computes its estimate using the values of the area of every environment we have in the dataset. From these values we extract the average area of the whole dataset, $\widehat{A}$, and we use it as our prediction of the total area of the environment. Then, we use this value to compute the predicted percentage of the environment we have explored.

Differently from the bounding box estimator, we cannot bound the error of the estimate of the mean area estimator. Therefore, it is not possible to

state if the prediction will underestimate or overestimate the actual area of the environment. Moreover, the mean estimator, although being a simple method to infer a prediction on the total area of the environment, is not robust to the presence of outliers in the dataset.

Based on these observations, we can state that the mean estimator is expected to work well when the dataset presents a low value of standard deviation on the areas of the environments. In this case, even if the mean is a biased estimator, the methods yields a prediction close to the actual value. This gives us another point to focus our research only a specific type of building (whose representatives are supposed to have less variance with respect to the mean), instead of a bigger heterogeneous dataset.

4.2.4 Area estimator from prediction error

The last model we present is the *area estimator from prediction error*. This model, instead of computing at the beginning of the exploration an estimate of the total area of the environment, computes iteratively a prediction of the area during the exploration. In order to do so, the model simulates the error of predicting the area, at any step, made by a generative model which estimates the total area of the building given the part of the building already explored, using the method introduced in [15]. Since the model presented in [15] requires, in order to infer knowledge on the explored environment, a complete semantic map of it, and since building a complete semantic map of the environment lies outside the scope of this thesis, we approximate its behavior by using a model that, given the explored area, produce an estimate of the total area of the environment. This estimate is similar to the one that can be computed with the generative model of [14].

Specifically, the model takes the information extracted from some sampled exploration runs in the dataset. From these, it computes the absolute error $\epsilon(p)$ (where p is the percentage of area already explored), between the actual area of the environment and the estimated area by every sample with the aforementioned method. These results are then used as input feature vector for a neural network regressor, which outputs the predicted error , $\hat{\epsilon}(p)$, over the exploration run for the whole dataset, composed by the prediction of the total area of a building obtained with the method of [14].

We choose as our neural network model a Multi-Layer Perceptron (MLP).

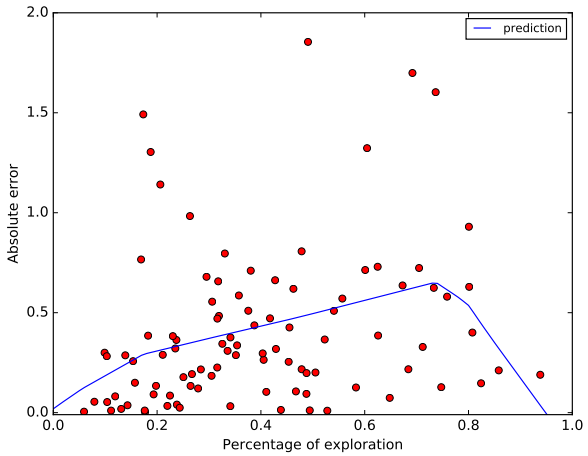


Figure 4.2: Prediction of the absolute error over the estimated area using a neural network

This model is formaed by a defined number of single perceptrons disposed on different hidden layers. We choose to use the implementation given by the *scikit-learn* Python library, which is presented in [19]. For regression problem with MLP, every perceptron learns a target function $f(x)$, starting from a set of training examples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, and outputs the prediction for an unknown input x in the form $\hat{y} = f(x)$ (because of the fact the problem is a regression problem, the activation function is the identity function). More specifically, we choose the implementation of *MLPRegressor* proposed in [27], which describes an MLP regressor with a single hidden layer of 100 perceptrons. We choose as our optimization algorithm Limited memory BFGS (L-BFGS), which uses the approximation of the inverse of the Hessian matrix (i.e., the second-order partial derivative of a function) to perform parameters updates. The choice is made because it is stated in [27] to converge faster and perform better with limited datasets with respect to the others optimization algorithms proposed in *scikit-learn* library.

In Figure 4.2, we can see the result of the regression made on the dataset we consider, using training samples in the form $(p_i, \epsilon(p_i))$ and a target function which is $\hat{\epsilon}(p)$. Then we use the target function learned during the simulation in the following way. At every time step t, we compute the estimate of the area of the environment, sampling from an uniform distribution

$\mathcal{U}(a, b)$, whose support is:

$$\begin{aligned} a &= \max[ExplArea, (1 - \hat{\epsilon}(p)) \times A] \\ b &= (1 + \hat{\epsilon}(p)) \times A \end{aligned} \tag{4.3}$$

where $ExplArea$ is the area explored until now and A is the target area. This value, $\overline{A_t}$, is then stored along with the last m estimates in the buffer $\bar{A}$. The actual prediction, $\widehat{A}$, is then computed extracting the median value of the stored estimates.

Since the estimate varies slightly during different prediction, we choose to store the last estimates and not only to return the latest one, because, in this way, we can filter out the error made during the sampling of the values, by smoothen the different predictions, which can be noisy, and using only their average. In order to reduce possible noise, we check for a possible switch every $T_s = \lfloor m/2 \rfloor$ cycles instead of every one. The choice of returning the median value can be explained in a similar way. Thanks to the fact that the median estimator is an unbiased estimator over a set of sample, it is expected to be lees prone to possible fluctuations derived from the sampling error we make using this model.

The area estimator from prediction error is expected to perform well on the whole dataset, assuming that we use a suitable number of predictions when we compute the median, in order to remove as much as possible the noise of the sampling.

Chapter 5

System architecture

In this chapter we present the architecture of the system, in terms of modules we have implemented for our research. We start by briefly describing MRESim, which is the simulator we use to develop the system. Then, we overview the main module implemented to enable the switch of communication strategies.

5.1 MRESim

MRESim is a simulator that has been specifically developed for multirobot systems employed in an exploration task. The simulator has been proposed and widely discussed by its authors in [31]. Here, we present a brief overview of the system in terms of its main modules.

MRESim is a discrete time simulator which works from a two-dimensional grid representation. At each time step, the simulator moves the robots to their new positions, updates their sensor information, simulates communication between the robots, and outputs the current state of the simulation. All the robots process their next step (i.e., the next position they move) simultaneously. Since this tends to be the most computationally intensive part of each time step, planning robot movements in parallel allows the simulator to take advantage of multi-core CPUs. Other events at each time step are processed sequentially.
Environments in MRESim can be uploaded from text-based or PNG files. The environments are occupancy grid models, with each cell being either free or an obstacle.

Algorithm 1 Simulation cycle of MRESim, taken from [31]

```
procedure SIMULATIONCYCLE(R)
    for all r_i ∈ R do
        while distance_moved(r_i) < max_speed(r_i) do
            nextStep = r_i.takeStep();
            if isValid(nextStep) then
                move(r_i, nextStep);
                                                    ▷ simulate sensing
                rangeData = findRangeData(r_i, nextStep);
                r_i.receiveRangeData(rangeData);
            else
                r_i.setError(True);
                break;
                                                ▷ simulate communication
    for all r_i ∈ R do
        for all r_j ∈ R, i ≠ j do
            if isInRange(r_i, r_j) then
                r_i.receiveMessage(r_j.createMessage());
                r_j.receiveMessage(r_i.createMessage());
```

5.1.1 Simulation cycle

In Algorithm 1 it is presented the pseudo code for a single simulation cycle of MRESim. The sequence of decisions is the following. Each robot decides where to move next, using the function *takeStep*. If the robot has a path to its destination, it keeps following it. Otherwise, it replans its path. The exploration strategy selects the best destination for the robot and generates a path. In order to be more efficient in selecting a destination, the exploration strategy can use all the information known by the robot about locations of its teammates, as well as any agreement made in previous communication with the BS or the other robots. This information is not globally available, but it is communicated when in range.

Once a robot has performed its step, the simulator provides sensor data from the new location. This sensor data are generated using raytracing from the robot's location at 1-degree intervals for a maximum of 180 degrees with a field of view centered around the current heading of the robot, simulating a real laser scan. A maximum sensor range can be configured for each robot,

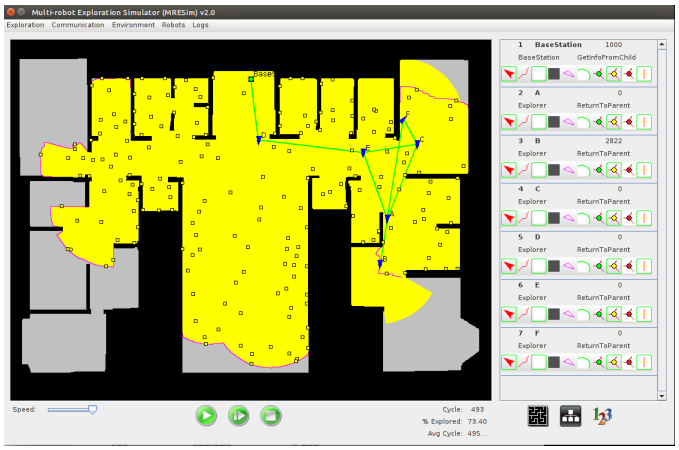


Figure 5.1: MRESim when the system is employing a MH strategy. We can note that the current frontiers at the BS (in purple) correspond to the area known by the BS (in yellow)

allowing for heterogeneity of sensors in the team. The robot, subsequently, turns these measurements into a polygon of free space (inner area), detecting the obstacles, detected as edges of the polygon. This polygon is merged in an occupancy grid. When two robots are within communication range of one another, they can decide to exchange their local maps in the form of occupancy grids.

Finally, the communication is simulated. MRESim supports a wide range of communication models, such as the ones presented in Chapter 2, which are static circle, Line-of-Sight, and an implementation of the Signal strength model. In principle, messages from both robots are exchanged only when they are sufficiently close, i.e., when the communication model predicts that the two robots are in range.

5.1.2 Realism of the simulation

MRESim does not take into account a number of factors that a multirobot system in the real world has to consider. The most important ones are:

- There is no sensor noise. In real settings, sensors can provide inexact data. For instance, laser scans can have spurious measurements either at maximum or at minimum range.
- Environments are two-dimensional and flat. In real scenarios this is almost never the case.

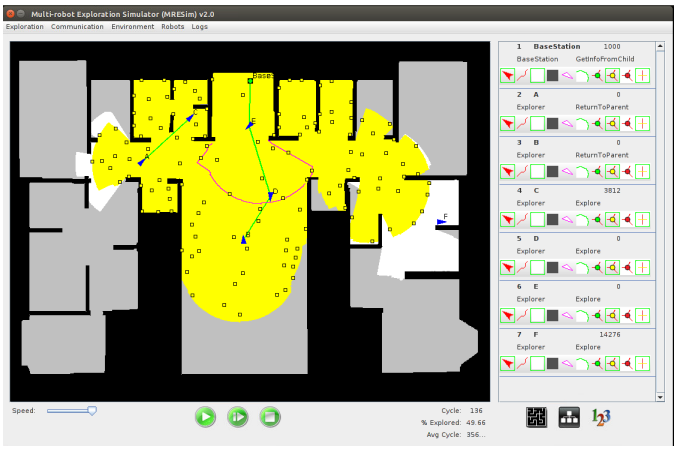


Figure 5.2: MRESim when the system is employing a RV strategy. The white area is the portion of the enivronment known by some agents but not already communicated to the BS

- The time is discretized. Real robots would run their own procedures, taking different amounts of time to do the processing and would not always move the same distance in each time segment.
- Communication is perfect. In real applications, communication is variable over time and it is usually difficult to predict.

Although having some drawbacks, MRESim is a reliable tool to compare results from different exploration algorithms, as it is stated in [31].

5.2 SwitchStrategy

We implement the switch of communication strategies inside MRESim. To do so, we develop a new strategy which works on the top of the ones already implemented. In MRESim, every exploration strategy is required to have two mandatory functions, which are *takeStep* and *replan*. The robots use these two functions to move in the environment and to plan the path to their goals, respectively. We develop a strategy in which the output of the planning is different based on the state in which the robots are.

In SwitchStrategy, every robot is assigned to a state. In every cycle, a robot can be in a *MultiHop* (Figure 5.1) state, in a *RendezVous* (Figure 5.2) state, or in a *Wait* state. These states are assigned by the BS at the beginning of the simulation cycle. Based on this fact, the *takeStep* function

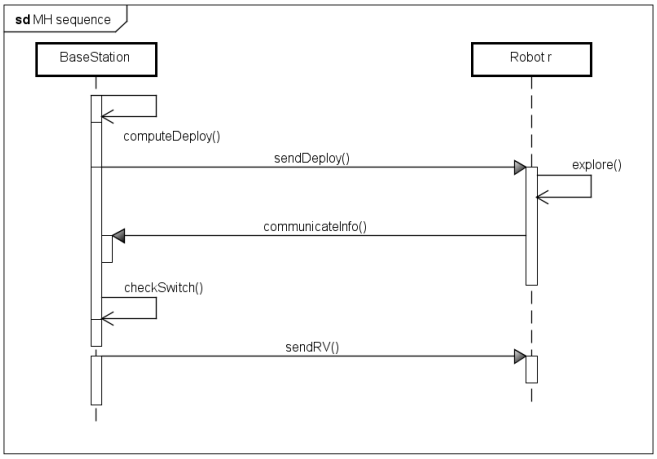


Figure 5.3: Switch from MH strategy to RV one

decides which of the already implemented functions the robot has to use. This means that in each cycle a robot can move using a MH strategy, a RV strategy, or it can perform no move.

Another important module of our implementation is the mechanism to switch between different strategies. This is done inside the simulation cycle by the BS, which is the only responsible of which communication strategy the team employs. After the simulated communication, the BS compares the two parameters, α and β, with the current exploration progress. If the conditions for a switch of strategy are met, the BS reopens the communication channel and sends its decision to the robots connected. In the case of a switch from MH strategy to RV strategy, shown in Figure 5.3, the BS alerts the robots to switch their state to *RendezVous* and, starting from the next cycle, the team employs a RV communication strategy.

The opposite case, from a RV strategy to a MH strategy, is shown in Figure 5.4. The BS alerts all the robots that are in its communication range to switch their state to *Wait*. The robots in this state stop their current task and move back to the BS until they are in communication range and, then, they do not move. The BS keeps sending this message in the following cycles, until all the robots are in range and in a wait state. When the team is finally reunited, the BS send a message to switch the state to *MultiHop*, allowing the replan of the deployment of the robots in a MH strategy starting from the next cycle. If the system employs the second communication channel, the BS broadcasts over such channel the signal to switch their states to

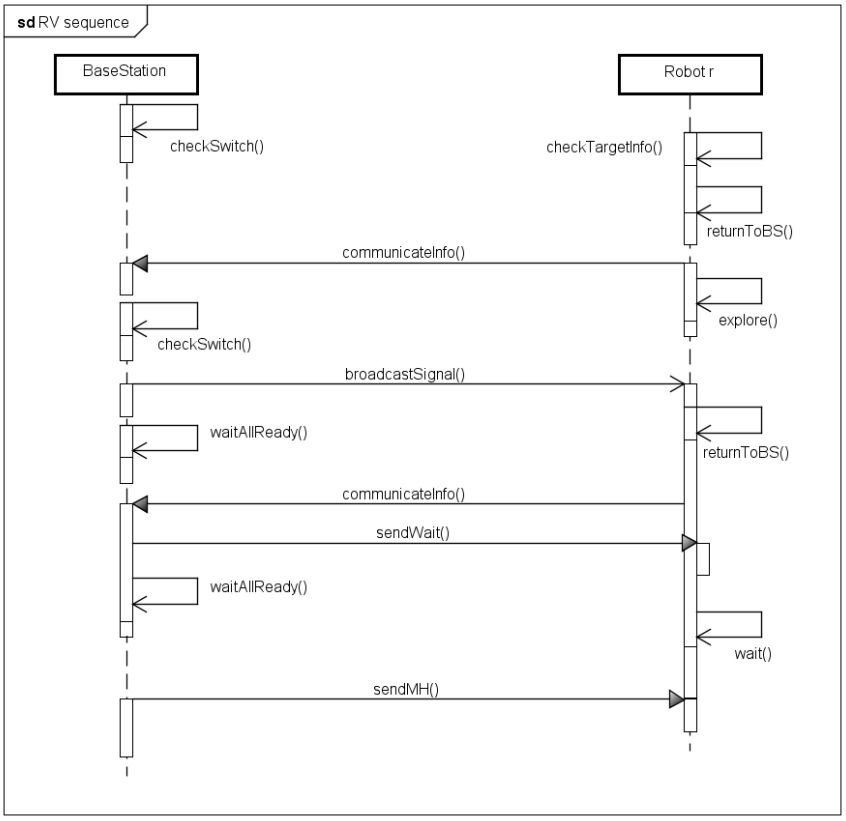


Figure 5.4: Switch from RV strategy to MH one

Wait. The robots go back to the BS and then wait that all members of the team regain connection.

Chapter 6

Experimental results

In this chapter we present the experiments performed in order to evaluate the performance of our approach. We start by illustrating the setting in which our experiments have been made. Then, we explain our evaluation procedure as well as the fixed and variable parameters in the evaluation.

6.1 Experimental settings

Among the different possible settings, we decide to use a multirobot simulator to evaluate the performance of our approach. The simulator we choose is MRESim, which has been introduced in the previous chapter. This choice has been made because, despite not giving a completely realistic simulation, MRESim is a well suited tool to compare different exploration strategies.

We choose to focus our attention on a specific type of environment, in order to comply with the prediction of the environment area made in Chapter 4. We decide to focus our attention on indoor environments, in particular school buildings, taken from the dataset used in [15]. The dataset contains the original floor plan of every environment. Since MRESim accepts only text files or PNG files with predefined dimensions as its environments representation, we convert the original floor plans in images of 800×600 pixels.

Based on this definition of the environment, we define our multirobot system. We have a team composed by a fixed number of robots, which is 6, and a Base Station. The robots have simulated laser scan sensors, whose data are given in input to the SLAM modules (which is explained in Section

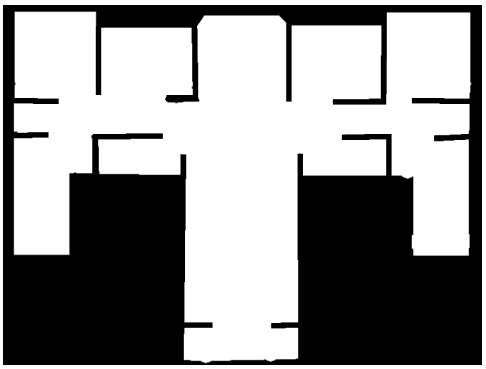


Figure 6.1: Environment westasheville

3.2), with a range of 90 pixels and a field of view of 120 degrees. The BS, on the contrary, is not equipped with any sensors and relies on the robots to get information from the environment. The robots and the BS are also equipped with two simulated RF communication systems. The first, which is the one the agents use to communicate standard information, has a maximum range of 90 pixels and employs a Signal strength communication model, which is constrained to a LoS model with the same range when the MH strategy computes the deployment of the robots. The second communication channel, which is the one the BS uses to broadcast the regroup information, employs a Static Circle communication model with a range of 800 pixels, in order to be sure to reach all the robots at the same time.

Given the fixed maximum size of the environments in the simulation, this setting is static throughout the experiments. In real cases, however, given the different dimensions of the environments, the same values in pixels can represent sensors and RF transmitters with different ranges in real environments. We do not consider this fact a limitation of our results, because the goal of these experiments is to investigate a conjecture made in [1] that is given based on a general multirobot system.

During the exploration we keep track of the information logged by the robots and the BS. Firstly, we log the data about the percentage of area explored by each robot, as well the percentage of the area known to the BS at each time step., at each time step. We use this information to evaluate the time taken to fully provide knowledge of the entire environment to the BS, for every simulation run. Secondly, we keep track of the distance traveled by the robots at each time step. At the end of exploration, the total distance traveled by the system is the sum of the distance traveled by every

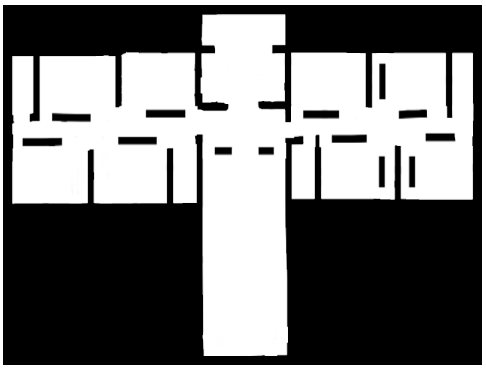


Figure 6.2: Environment indianhead

robot. At last, we log the incremental time not in communication with the BS for every robot. This is defined as it follows. At any time step, if the robot is not in communication with the BS (either in direct communication or by means of other teammates which serve as relay), its time not in communication is incremented by one. At the end of the exploration, we want to analyze the mean time not in communication at every time step.

6.2 Evaluation procedure

Given the fact that we consider the metrics discussed above, which are the time taken to explore the environment, the time not in communication with the BS, and the total distance traveled by the robots, we decide to structure our evaluation procedure in the following way.

The main question we should answer from a design point of view is which value of α and β are best suited for a particular environment. We choose a set of pair (α, β) taken from the interval $(0; 1]$ with a fixed step of 0.1. This procedure is employed for all the different prediction models, including the perfect oracle. This method is then replicated using the second communication channel.

We present the result using heatmaps, in which every pair of α and β corresponds to a point in the heatmap. We group the simulation runs based on the environment and the prediction model. We, also, normalize the results in order to extract comparable figures between different environments. We use a conventional color code for our heatmaps, using a gradient that varies from green to red, through yellow. The meaning is that the results

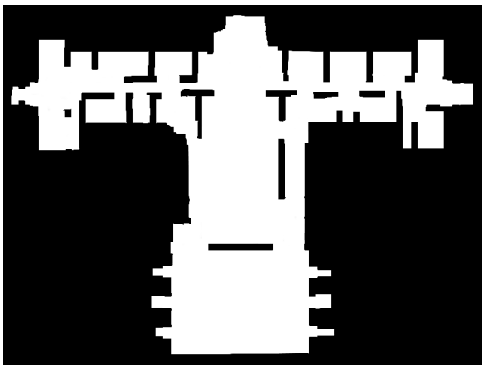


Figure 6.3: Environment henderson

in the green area are the best for that metric, while a red color means that the performance is negative with respect to the other results on that environment. We also extract average performance of the strategy, combining all the different environments.

The normalization procedure of the results works as follows. For every performance metric we collect the logged data from the simulations on every pair (α, β). Then, we compute the maximum and the minimum value among all the prediction models and normalize the performance for any pair of parameters in the interval $[0; 1]$ in the following way:

$$r_n = \frac{r - min(R)}{max(R) - min(R)} \tag{6.1}$$

where $R = \{R_P, R_M, R_{BB}, R_E\}$ represents the set of all the results for every pair (α, β) with every model for a given metric, and $r \in R$ is the results for a given pair (α, β). Then, the average performance for all the environments is computed in a similar way, summing the results for every environment explored and then normalizing them in the same way.

6.3 Parameters

Before starting any simulated exploration run, the system needs to receive several inputs to set some fixed and varying parameters in MRESim. Most of them are used to set thresholds required by the communication strategies the system employs. The fixed parameters are:

- **Target Information Ratio**: as we already presented in previous chapters, this is used by the RV strategy to decide when the robots

should go back after they explore a portion of the environment. In [30], the authors show that the performance of a system employing this strategy are better or at least comparable with the performance of an explicit RV strategy. We choose a value of *targetInfoRatio* of 0.80 because it is a good balance between pushing the exploration farther and maintaining a periodic connection with reasonably short time interval.

- **Replanning Threshold**: this parameter is meant to be used when the system is employing the MH strategy. It represents the minimum number of robots in ready state (i.e., that have reached their destinations) that the BS has to wait before replanning. We choose to set this parameter to 1, meaning that the BS computes a new deployment as soon as only one robot is ready. This value has been chosen based on the observations in [5], where the authors state that, with this replanning threshold, the system has the best performance.

- **Communcation parameters**: here we group all the parameters related to the Signal strength model and the LoS model introduced in Section 3.2. For the Signal strength, as reference distance d_0 we choose the maximum distance at which two agents can communicate, which is 180px. The signal strength at the reference distance P_{d0} is equal at -92 dB. Our path loss factor N is equal to 2 and the maximum number of walls C is equal to 4. Finally, we choose a wall attenuation factor WAF of 3. For the LoS model employed during the MH strategy, we choose as maximum distance d for the communication the maximum range at which two agents can communicate, which is 180px.

Moreover, for every simulation we have a pair of parameters, which are α and β, that vary from a simulation run to another. We decide to model this behavior imposing that these pairs must have a feasible assignment, $\alpha < \beta$, reflecting the conjecture proposed in in [1] where the MH strategy must come before the RV strategy. We choose to have values that can only vary with a fixed quantity. We use a fixed step of 0.1, because it gives us a good flexibility in terms of coverage of the values in the interval $(0; 1]$, without having too much simulations to run.

With the area estimator from prediction error we add two more parameters which are the number of samples m used to compute the median and the time interval T_s between two consecutive check for a switch at the BS. We fix the time interval as half of the buffer size for every simulation, $T_s = \lfloor m/2 \rfloor$.

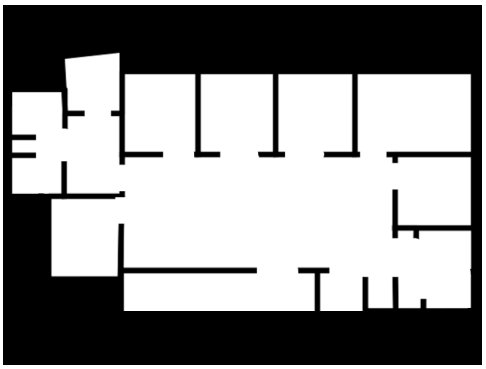


Figure 6.4: Environment scuola

6.4 Results

In this section we present extensively the results obtained by our method. Before doing this, we contextualize some important definitions about the chart we display. As it was stated before, our results will be presented using heatmaps, which have a color convention to represent the performance of the system according to a pair of parameter α and β. In the heatmaps, we can find important points which can be used to have a better understanding of the results. In the top left corner, we find the performance for the simulation with $\alpha = 0.1$ and $\beta = 1.0$, which can be seen as a full RV strategy, since the first switch is extremely close to the beginning of the simulation and the second one is never employed. On the bottom left corner (and more in general on the diagonal), we find results where the pair of values is in the form $\beta = \alpha + 0.1$. These simulations have the same behavior of a full MH strategy, because the parameters have similar values and the RV strategy behaves more like a few random steps in the environment, rather than a structured exploration strategy. We state that, whichever prediction model we use, the pair $(\alpha, \beta) = (0.1, 1.0)$ and pairs $(\alpha, \beta) = (k, k + 0.1), k \in [0.1; 0.9]$ are the performance bounds of our strategy.

6.4.1 Perfect oracle model

We start to present our results showing the performance of the system when the switch is driven by a model where the predicted area of the unexplored environment is the real area. This means that the estimate gives no error.

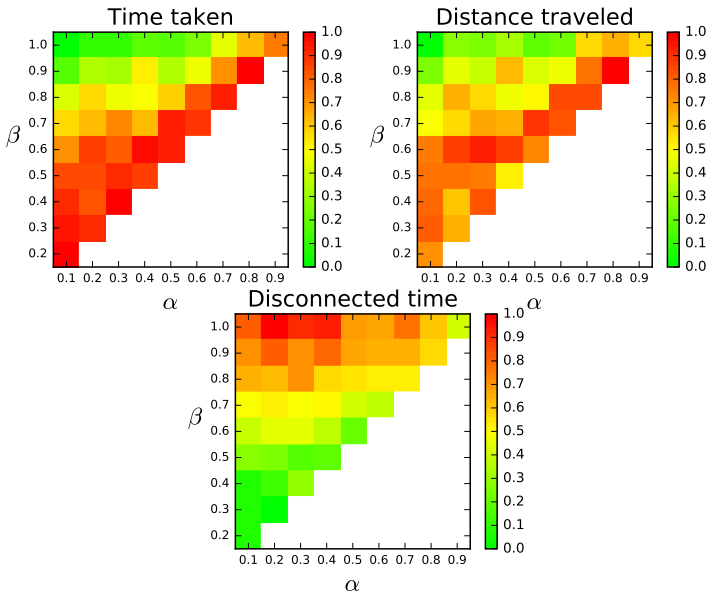


Figure 6.5: Average performance of the perfect model

Although not being applicable in settings where it is not possible to know the area of the environment a priori, it is useful to test the performance of the system against possible different communication strategies. It also gives us baseline results which can be useful to compare the different prediction models we employ.

In Figure 6.5, we present the average results for the three performance metrics we take into account for the perfect oracle. The first thing we can note is that we have two conflicting results to optimize, as we already said in Chapter 3. These are the time taken to explore the environment and the time not in communication between the robots and the BS. We also note that the distance traveled seems to follow the same behavior of the time taken to explore the environment.

Focusing on the time taken to explore the environment and on the distance traveled, we note that the best performance is provided by strategies where there is a notable difference between the values of α and β. We can say that the best results come in the interval of $\alpha \leq 0.5$ and $\beta > 0.7$. We view these results as a consequence of the fact that, for these values, the RV communication strategy is dominant, allowing the robots to explore the

environment faster than using the MH strategy to communicate.

The best performance and our lower bound is indeed the pair of values $(0.1, 1.0)$, while the worst ones (i.e., upper bound) are when the difference between the parameters is minimal, $(\alpha, \alpha + 0.1)$. This prove our hypothesis that the performance of our approach is in the middle between a pure MH strategy and a pure RV one.

Analyzing the second performance metric, which is the distance traveled by the group of robots, we find that it is somewhat similar to the time taken to explore the environment. This cannot be considered a surprising result, because the more time the system needs to explore the environment, the longer the robots have to travel. As a consequence, that trying to minimize, with our composed strategy, either the time taken to explore the environment or the distance traveled, affects in a positive way the other metrics.

Focusing on the last metric, which is the time not in communication between the robots and the BS, we find a complete different scenario. Here, the results obtained are the inverse of those relative to the time taken to explore the environment, which means that we cannot optimize both of these performance metrics, but we should find a trade-off between the two.

This conflicting nature is widely observable if we focus our attention on the upper and lower bound on this last performance metric. The best results are obtained with lower values of both α and β. This behavior reflects the fact that the strategy that optimizes the time not in communication is the MH strategy, where the communication is enforced to be almost continuous.

In Figure 6.6, a comparison is made between a system with the second communication channel (Figure 6.6b) and a system without it (Figure 6.6a). On the top the average time taken to explore the environment is reported, while on the bottom it is reported the time not in communication between the robots and the BS. Focusing on the latter one, we can notice that the second communication channel gives the system a slight, but significant, reduction in the time the robots are disconnected.

This behavior happens because, when the BS broadcasts the alert message to regroup to the robots,they immediately go back to the BS and, by doing so, minimize the time required to establish again a complete connection among the while group. Such improvement, given by the second

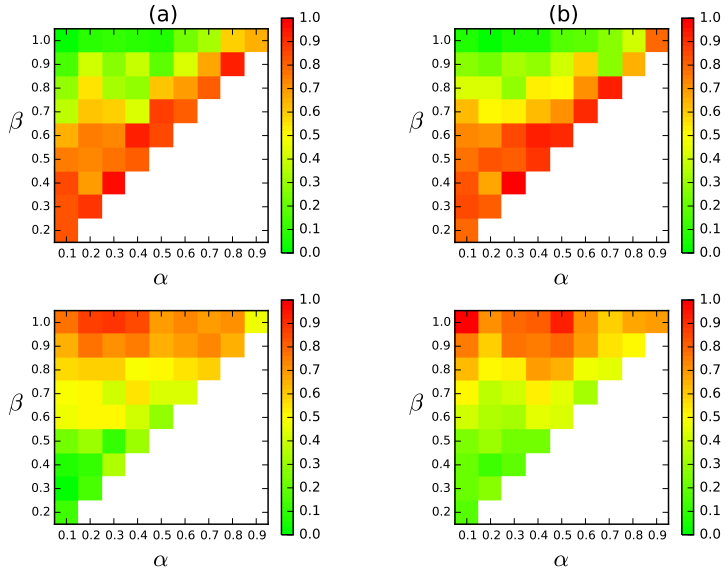


Figure 6.6: Comparison between time taken to explore the environment (top) and time not in communication (bottom), using the second communication channel (b) or not (a)

communication channel, seems to be greater when parameters are in the range $\alpha \leq 0.4$ and $\beta \leq 0.7$.

Moreover, using a second communication channel in this way seems not to have particular influence on the time taken to explore the environment. As a matter of fact, the results with or without the second communication channel are almost identical. Given these facts, it seems that using a second communication channel does not affect the performance of exploration, while reducing the average time the robots are disconnected using a better regrouping capability.

6.4.2 Bounding box estimator

The second model we investigate is the bounding box estimator. As we stated in Section 4.2.2, this prediction model is the simplest we utilize and it is essentially an overestimate of the actual area of the environment. Due to this fact, the performance can be completely different from the perfect oracle.

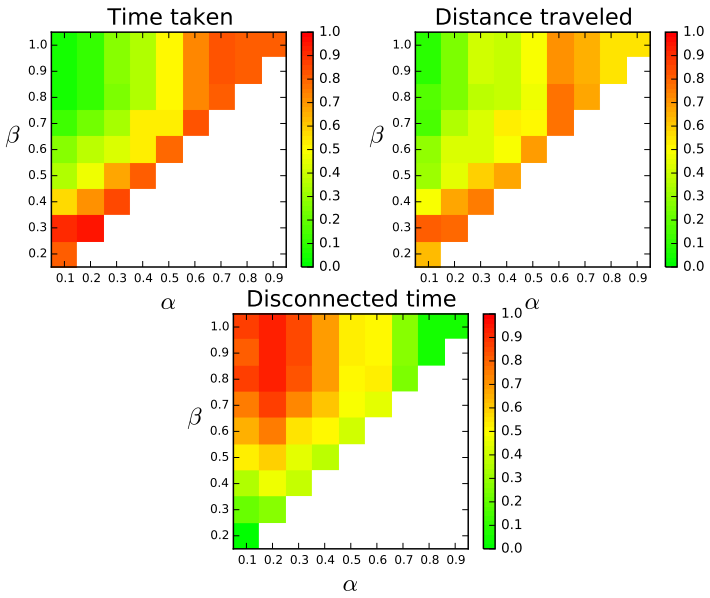


Figure 6.7: Average performance of the bounding box model

In Figure 6.7, we show the average normalized performances of the bounding box model. Focusing on the time taken to explore the environment and the total distance traveled by the team, we can note that they behave in a similarly to each other and minimizing one of them yields good results also for the other. The first observation we can state is that, in this case, the area in which the system yields better results (namely, the green area) seems to be larger with respect to the perfect oracle model. Also, this area is slightly moved from values where the RV strategy is predominant (for instance the strategies with α low and $\beta = 1$) to pair of values when $\alpha < 0.5$ and $\beta \geq 0.6$.

These results seem to confirm that the bounding box model performs good with combinations of parameters. Consider that the bounding box model overestimates the environment. This overestimate is translated, in terms of switch of communication policy, to an overestimate of the switching parameters. In this case, when an exploration run is approaching to discover the whole totality of the environment, the estimator can state that the percentage of the environment explored is much smaller than i actually is. This leads to never employ the second MH strategy, because the threshold β is never met by the estimated percentage of the area.

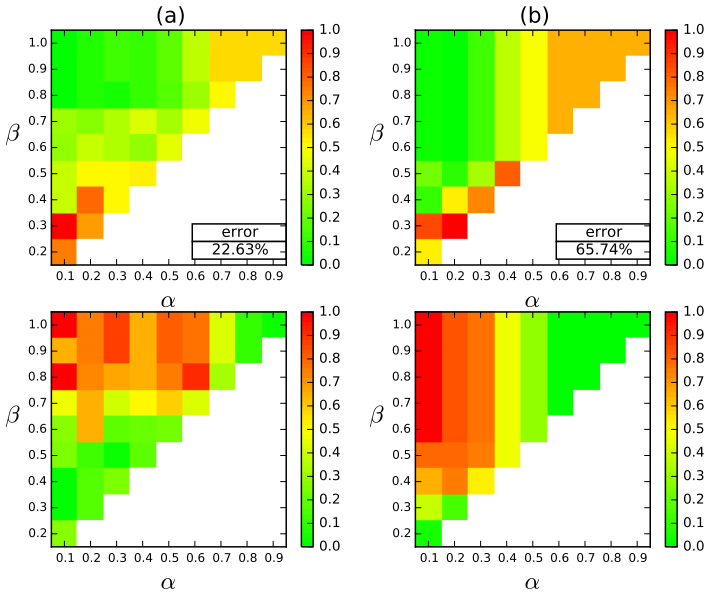


Figure 6.8: Comparison between time taken to explore (on top) and time not in communication (on bottom) of two environments. (a) is Figure 6.4 and the bounding box estimator slightly overestimates the actual area. (b) is Figure 6.1 and the bounding box estimator is less efficient in predicting the actual area

Focusing on the time not in communication, we can observe the same problem of the other metrics. The best results are yielded in the top right corner and in the bottom left one. This follows the previous observation on thresholds that are never reached. For instance, the best performance is yielded with $\alpha = 0.9$ and $\beta = 1$, because this strategy, using the bounding box estimator, is behaving as a pure MH strategy, in which the time not in communication is always optimized.

With the bounding box estimator, finding a suitable trade-off, between the time taken to explore the environment and the time not in communication between the robots and the BS, seems to be very difficult, because the pairs of parameters that have suitable performance for all the metrics are hard to find.

In order to confirm this hypothesis, we investigate the difference that can be found in a set of different single environments. In Figure 6.8, we compare the results on two different environments with respect to the time

taken to explore the environment and the time not in communication. In Figure 6.8a, we have an environment (Figure 6.4) where the bounding box estimator seems actually able to give a good prediction of the actual area. In Figure 6.8b, the model overestimates the actual area of the environment (which is presented in Figure 6.1).

We can notice that the results seem to be very different in the two environments. In particular, it seems that, the more the prediction overestimates the total area of the environment, the more freedom we have in selecting values for the parameters without penalizing too much the time taken to explore the environment. When we analyze the time not in communication between the robots and the BS, it seems that we have different results based on how much the prediction is wrong. With a bigger error, the best results are given by values of $\alpha, \beta \geq 0.5$, while, when the error is smaller, the heatmap resembles the results of the perfect model. This behavior seems to confirm the hypothesis that the overestimate of the predicted area leads to a set of scenario where the parameter β is never reached, causing better performance related to exploration time, but negative one in maintaining connection with the BS.

Furthermore, we want to investigate wether the peculiar features of the environment can affect, rn not, the prediction obtained using the bounding box estimator. We choose to analyze the shape of the environment because it is the most important feature the model takes into account.

In Table 6.1, we present the average prediction error of the bounding box model on different shapes of environments. The results show that, as long as the environments have a rectangular shape, the bounding box model returns a prediction of the area that is a slight overestimate of the real value, and so we can consider this estimate as reliable. When the shape of the environment starts to be more complex, the error of the prediction increases and the estimate becomes less reliable. This can lead to say that the bounding box model is a reliable estimator only if we have rectangular-shaped buildings because, in these situations, it tends to return a slight overestimated prediction of the actual area of the environment.

	Shape		
	Rectangular	T-like	H-like
Avg. error (%)	28.41%	102.54%	48.74%

Table 6.1: Average prediction error for different shape

6.4.3 Mean area estimator

The next model we investigate the performance of is the mean area estimator. Recalling what we have said in Chapter 4.2.3, the mean area estimator gives a prediction of the actual area of the environment computing the mean area over a dataset. This type of estimate is certainly affected by the outliers of the dataset, but it is fairly simple to compute and implement.

In Figure 6.9, it is presented the average performance in terms of time taken to explore the environment, distance traveled by the robots, and time not in communication between the robots and the BS. Focusing on the time taken, the behavior we notice is that, on average, the system seems to have better results when parameter α has a low value, while β is close to 1.0. The worst performances, on the other side, are when the system employs a strategy similar to a full MH one (i.e., the pairs of values on the diagonal). These results seem similar to those of the perfect oracle and they seem to confirm the goodness of this model in predicting the area of the environment. Considering the distance traveled we can see that the area where the system performs better resembles that in which the time taken to explore is better.

With respect to the time not in communication between the robot and the BS, we find that the system has good performance with most of the values for the parameters. Specifically, the he set of experiments where the performance is worst is very limited to situations where α has a low value (≤ 0.4) and β is exactly 1.0, that is where the system behaves more similarly to a plain RV strategy, instead of a mixed one.

With this specific model, it seems that it is easier to find values that can represent a good trade-off between the need to explore the environment as fast as possible and the need to maintain a connection as stable as possible. Without being the best in any of the metrics, values like $\alpha = 0.4$ and $\beta = 0.8$, seem to reach good performance in both of them, sacrificing some speed in exploring the environment to maintain a better connection between the robots and the BS, and *vice versa*.

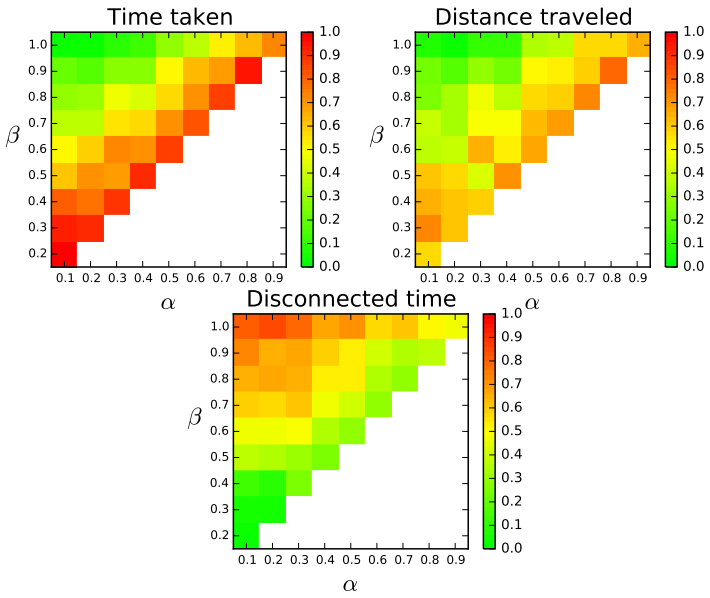


Figure 6.9: Average performance of the mean area estimator model

Furthermore, given the nature of the model, we want to analyze performance obtained on specific environments in extreme cases. The aim is to verify how much the model can be trusted when the estimate made turns out to be substantially wrong. In Figure 6.10, a comparison is made between different exploration runs on two environments. Figure 6.10a shows a setting (Figure 6.2) in which the mean area estimator is actually an overestimate of the real area of the environment, while in Figure 6.10b, the model underestimates the actual value of the area of the environment (which is presented in Figure 6.3).

Comparing the results found, we observe different behaviors. When the mean area estimator returns a value that overestimates the real value of the area of the environment, the results seem to be similar to those of the bounding box estimator. A significant number of different pairs of values of the parameters returns performance closer to the best one, if we consider the time taken to explore the environment. When we take into account the time in disconnection with the BS, the majority of the pairs analyzed yield an overall negative performance. This behavior seems to recall the bounding box estimator and they are actually quite similar. In both cases it happens

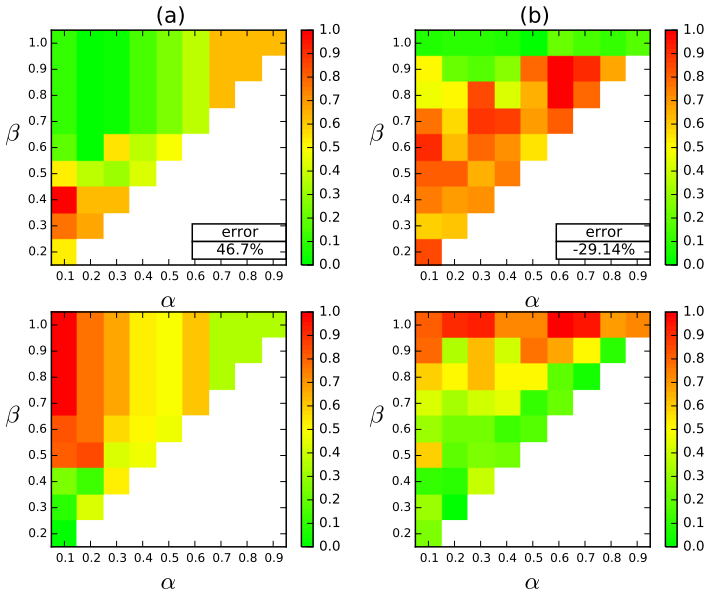


Figure 6.10: Comparison between time taken to explore (top) and time not in communication (bottom) of two environments (Figures 6.2 and 6.3), using mean area estimator

that, due to the fact that the prediction is an overestimate of the actual area, the system never switches from the RV strategy to the MH one, resulting in faster overall exploration, but losing connectivity with the BS.

On the contrary, when the model underestimates the actual area of the environment we have a completely different behavior. In terms of speed of exploration, the best results are essentially with pairs of parameters where $\beta = 1.0$, while the others are generally negative. Considering the time not in communication between the robots and the BS, the result is the opposite. Only simulations where $\beta = 1.0$ have a negative performance relatively on this metric, while pretty much all the other pairs yield good results. The behavior can be explained as follows. Because of the fact that the mean area estimator gives a prediction that actually underestimates the real area, the system anticipates the time of the switches. As a matter of fact, the switch from the RV strategy to the MH one happens before it is supposed to happen and this fact leaves more time to the robots to operate in MH strategy, allowing the whole team to be more connected with the BS. The only simulations that are not affected by this behavior are the ones when the RV strategy is not required to switch back to the MH one, because the parameter β is set to 1.0.

The considerations made on single environments shows that the mean area estimator seems to be a reliable model on average, which has similar performances to the perfect oracle in terms of time taken to explore the environment and disconnection with the BS. In most of the cases, it seems that is possible to find a good trade-off between the two metrics. On the other side, the model seems to show its weakness when the environment it deals with a completely different size with respect of the dataset analyzed.

6.4.4 Area estimator from prediction error

The last model we analyze is the area estimator from prediction error. This model differs from the previous one because it does not return a fixed value at the beginning of the exploration. Instead, it computes, at each time step, a new estimate of the total area of the environment using a prediction of the absolute error made in estimating the area.

To do so, we star from a method that is able to predict the total area of the environment given a partial exploration, [14]. Such method requires a semantic map of the environment to compute the prediction of the area. In order to approximate the application of such method without the burden of having to calculate the complete semantic map, we model the performance of the method while estimating the area. This is done, as explained in Section 4.2.4. by fitting a neural network, which, from a set of inputs of punctual absolute error on a given percentage of exploration, $(p, \epsilon(p))$, extracts a model of the error $\epsilon(\hat{p})$ that can, reasonably, be performed by the method [14] on estimating the area. Then, it predicts the value of the area sampling with this error in an uniform interval around a goal area and returns the actual estimate computing the median on the last m samples.

Figure 6.11 presents the average normalized performance for the error estimator computing the median on the last 70 samples. What we can observe from the results is the following. If we analyze the time taken to explore the environment, there seems to be a clear distinction between the pairs of parameters that perform negative and the ones whose performance is good. We could place the separating line on $\beta \simeq 0.6$ and state that every combination of parameters (α, β), with ≥ 0.6 has performance close to the best one. This result could be explained as follows. Using the median on the last m samples along with the fact that the system checks if it has to

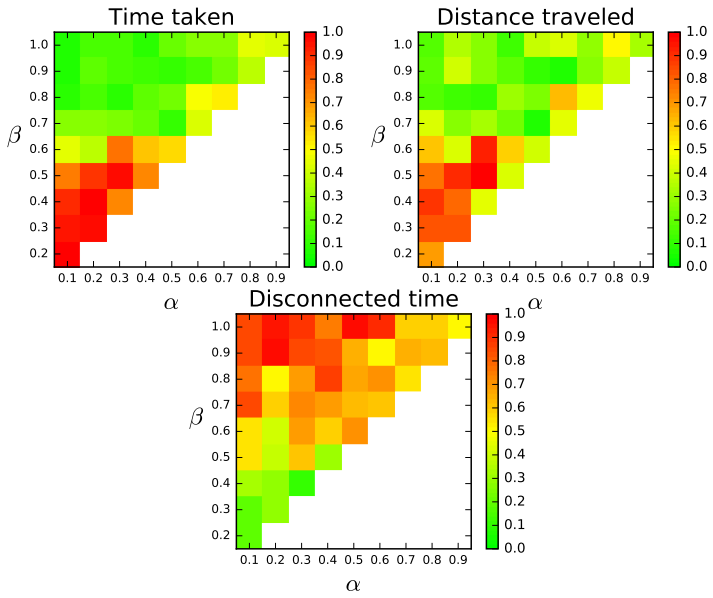


Figure 6.11: Average performance normalized for the area estimator from prediction error using the median on the last 70 samples

change communication strategy every T_s cycles instead of every time step, creates a system that is less reactive in adapting to a possible switch of communication strategy.

Considering the time not in communication between the robots and the BS, the same behavior observed before appears. In fact, also with this metric, it is possible to quite clearly bound the set of simulations in which the system has the best performance against the one in which the system has the worst one. Here we can place the separation at $0.5 \leq \beta \leq 0.6$. All the pair of values above this line have negative to worst performance in terms of communication capabilities, while the others yield good results. This seems to reinforce the hypothesis above, i.e., this setting results into a system less reactive on possible changes of the communication strategy.

In order to further investigate this issue, we run simulations where the length of the buffer in which we store the samples varies and we compare their normalized results. Figure 6.12 presents a comparison between a prediction error estimator with a median on 70 samples (Figure 6.12a) and a median on the last 30 samples (6.12b). Considering the time not in communication, when the model uses less samples to compute the predicted area

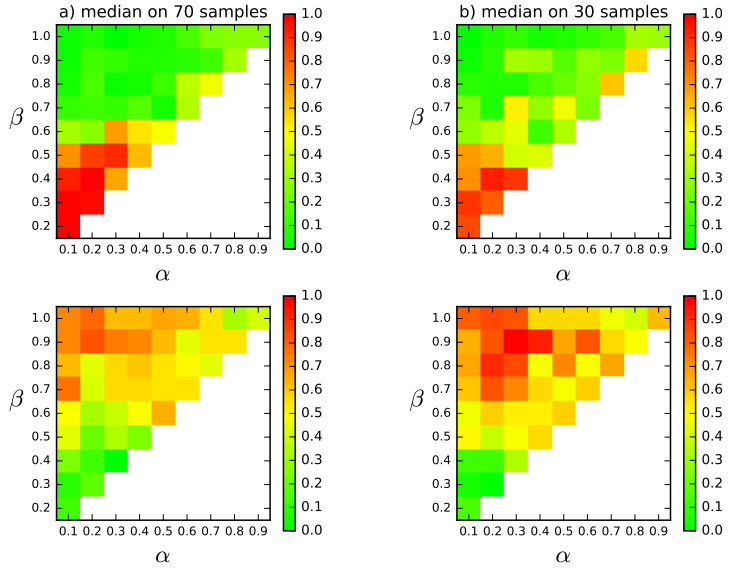


Figure 6.12: Time taken to explore the environment (top) and time not in communication (bottom) for different sample's buffer size, using area estimator from prediction error model

of the environment, the system has slightly worse performance with respect to the situation when the buffer size is bigger. The line delimiting the good and negative performance is similarly placed ($\beta \simeq 0.4$). Considering the time taken to explore the environment, it seems that using a larger buffer to store the samples results in slightly better performance overall. Even if using a smaller buffer seems to help on lower values of parameters, the best performance in this metric are given by a larger size buffer.

This pattern seems to be confirmed throughout the experiments. As a matter of fact, it seems that reducing the size of the buffer affects the performance too. As we include less samples in the buffer the performance degrades until a limit point in which the prediction becomes noisier and it might happen that consecutive predictions yield opposite results. In that case, the system is stuck in a loop. For instance, considering the limit case of a single element buffer and checking for the switch at every time step, we could have situations in which the system keeps switching strategy between MH strategy and RV one and *vice versa*, because the predicted area is floating around one of the switch parameter. This loop can eventually be resolved or continue.

Adding samples to the buffer and computing the median, along with checking for the switch not each time step, seems to partially solve this issue, but at the same time it introduces the drawback of a system less responsive to the switching condition. Moreover, computing the median is useful to filtering noisy data, but, at the same time, if the error is high (like in some of our simulations), limiting the effect of a number of mispredicted samples becomes more difficult. Therefore, it seems necessary to find a suitable number of samples that can work as a trade-off between avoiding possible loops and using too many samples. We find that in our settings a good choice is between 30 and 70 samples in the buffer with checking for switch every 15 to 35 cycles.

The combination of this model and our policy seems not to work completely flawless. Using the percentage of area explored to choose when the system has to switch between different communication strategies, along with a prediction of the total area of the environment dependent to exploration progress might lead to the aforementioned issue with never-ending loops.

6.4.5 Models comparison

After having presented the performance obtained with all the models, we want to compare them to see how they perform against each other. We use as baseline for the comparison the perfect oracle. Doing this, we can evaluate how much the models differs from the perfect setting.

We start by comparing the perfect oracle model with the bounding box estimator. Looking at Figure 6.5 (perfect oracle) and Figure 6.7 (bounding box estimator), it seems that using the bounding box estimator gives the advantage of having more pairs of values of parameters that perform better with respect to the perfect model when the need is to be quicker in exploring the environment. Because of the fact that the model returns an overestimate we are less constrained in choosing α and β to have a good performing system. When we take into account the time not in communication and the need to maintain the robots connected with the BS, this model seems to show its weakness, performing significantly worse than the perfect oracle, due to the overestimate of the area of the environment. This model has an edge, however, when the environments has a rectangular dense shape, like shown in Figure 6.8a, where it seems to yield a similar, if not better,

performance than the perfect oracle.

Considering the performance of the mean area estimator (Figure 6.9) against the perfect model, it seems that the mean area estimator has an overall performance that is similar to the perfect one. The area in which they perform well has a similar shape and spans relatively on the same pairs of values for the parameters α and β. Moreover, it seems that the mean area estimator can be a good choice if we want to have a trade-off between the speed in exploring the environment and the need to be connected to the BS. However, even if the general performance is similar to the baseline, the model shows its weakness in extreme cases. As a matter of fact, when the area of the considered environment is close to the estimated mean, the model predicts in a good way the real outcome, while, as it is shown in Figure 6.10, when the considered environment has an actual area that deviates from those present in the dataset used to train the estimator, the model has difficulties. Overestimating or underestimating with this model seems to be an issue to deal with.

When we consider the area estimator from prediction error (Figure 6.11) against the perfect model, different considerations can be made. Before starting comparing the models, we must choose the number of samples m that should be stored when making the prediction and the interval T_s the system uses to check if it has to switch communication strategy. The choice of these two additional parameters adds further complexity to the system and changes the behavior of the proposed policy. If the choice is not wise, we can experience issues with loops switches between a strategy to another and back. When a suitable number of samples and time interval are chosen, the results shows that this model seems to perform well considering the time taken to explore the environment, because the less reactivity of the whole system is translated in more freedom of exploration which results in faster exploration progress. The model might not perform good in terms of connection between the robot and the BS, where only a small number of pairs of values of parameters α and β maintain a connection of similar quality to the best result in the perfect model.

Comparing the models to each other, it emerges that each of them can address specific situations. The bounding box model seems to be useful when no data are available on the type of building and the robots cannot utilize previous knowledge such as dataset, but only aerial pictures of the area. The mean estimator seems to be a good choice when the system has a previous

knowledge in terms of type of building it is visiting. The area estimator from prediction error can be useful if the we can limit the variance of the samples and the noise on the prediction, resulting in a fast but less connected system.

Lastly, we want to summarize what we have found about the second communication channel. Throughout the experimental session we employ the second communication channel in addition to the normal one. The results, like in Figure 6.6, seems to show that, overall, using a second communication channel, to alert the robots of an imminent switch, has an edge in terms of improving connection capabilities. The robots stay connected for more time with the BS, while maintaining similar performances in terms of time taken to explore an environment. Given these facts, it seems that the second communication channel should be employed when there is the possibility. A possible drawback of employing it is that it could be an expensive addition to a multirobot system and therefore it cannot be applied to all the possible settings.

Chapter 7

Future work and conclusions

This thesis has investigated a possible solution to trade-off between conflicting objectives in multirobot systems that have to explore an unknown environment as fast as possible while trying to be as much connected as possible to a Base Station (BS) placed at starting position of the group. We followed the hypothesis suggested in [1], according to which, in such a multirobot system, different communication strategies work well in different phases of the exploration. Specifically, a Multi-hop communication strategy (in which robots have to maintain a continuous connection with the BS) should be employed at the beginning of the exploration, followed by a Rendezvous one (in which the robots are required to report to the BS only from time to time) in the middle of the task, and finally concluding with a Multi-hop strategy. After having formulated the problem a multirobot exploring system addresses, we designed a general policy in which the switch between two communication strategies is based on the exploration status. Then, we instantiated a solution in which the switch between the Multi-hop strategy and the Rendezvous one is done according to an estimate of the percentage of the area of the environment already explored. We modeled the unknown total area of the environment with a bounding box model, a mean estimator (calculated on a dataset), and a prediction of the error between the estimate of the area and the actual one during previous estimate. Moreover, we extended the system to work with a second communication channel, that acts as a synchronous broadcast channel for the group.

The experimental evaluation has been conducted using MRESim to simulate a multirobot system in an exploration setting. These experiments have been performed in numerous indoor environments taken from a dataset of

schools and have evaluated our prediction models against a perfect oracle. The results showed the following behaviors:

- The bounding box estimator seems to be a good method when no data are available on the type of environment and the robots cannot utilize previous knowledge such as datasets, but only aerial pictures of the area. Even if the predictions made are always an overestimate of the actual area, the method is not restrictive on the choice of the parameters. The possible drawback seems to be the negative performance on maintaining the connectivity of the team.

- The mean area estimator seems to be a good model when the system has some sort of previous knowledge on the type of environments it has to explore. To maximize its performance, a dataset as homogeneous as possible is required. Under this condition, this model achieves similar results as the perfect oracle model.

- The area estimator from prediction error might be useful if we are able to manage the complexity introduced by the samples' buffer and the time interval after which the switch is checked. If, with an accurate choice of the values for these two parameters, the system can limit the variance of the samples and the noise of the prediction, the model yields good results in terms of time taken to explore the environment. In general, however, the combination of a switch and a prediction both dependent on the exploration progress can cause failure in the system.

- Using a second communication channel can help the system to reduce the time not in communication between the robots and the BS, enabling faster switches between RV strategy and MH strategy. However, its deployment it is not always possible due to peculiarity of the environment or an excessive cost for the infrastructure.

Overall, the experimental evaluation has shown that a composition of communication strategies, although not being the best in any performance metrics, might be considered a good all-around communication strategy when the goal is to optimize multiple metrics, such as exploring the environment in less time as possible while trying to maintain a reliable connection to the BS.

In order to confirm our findings, further work is needed. We list here possible future works for our system.

- **Extension of the policy**: in this work we analyzed a specific policy, based on the percentage of the total area explored, for switching be-

tween different communication strategies. An extension, following the general function presented in Chapter 3, is possible, allowing to work with an arbitrary number of switches and with conditions different from the area of the environment. This could lead to a more general framework for mixed communication strategies.

- **Employing other prediction models for the total area of the environment**: the prediction models presented here are simple models used to verify our initial hypothesis. Different prediction models for the total area of the environment could be used and analyzed.

- **Extension on the team composition**: this thesis has extensively analyzed a system with a fixed number of homogeneous robots. Future work can explore different initial team configurations as well as heterogeneity among the team to split the exploration effort based on the capabilities of different agents.

- **Second communication channel**: future work can more extensively investigate the topic of using a second communication channel to make the team more responsive in switching, building on the preliminary results we have found.

Bibliography

[1] Torsten Andre. *Autonomous Exploration by Robot Teams: Coordination, Communication, and Collaboration.* Phd thesis, Alpen-Adria-Universität Klagenfurt, 2015.

[2] Ronald C. Arkin and Jonathan Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *Proc. 7th International Workshop on Advanced Motion Control*, pages 455–461, 2002.

[3] Alper Aydemir, Patric Jensfelt, and John Folkesson. What can we learn from 38,000 rooms? Reasoning about unexplored space in indoor environments. In *Proc. IEEE International Conference on Intelligent Robots and Systems*, pages 4675–4682, 2012.

[4] Victor Bahl and Venkat Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proc. 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 775–784, March 2000.

[5] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, Ioannis Rekleitis, and Francesco Amigoni. Asynchronous multirobot exploration under recurrent connectivity constraints. In *Proc. IEEE International Conference on Robotics and Automation*, pages 5491–5498, 2016.

[6] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, and Francesco Amigoni. Communication-constrained Multirobot Exploration: Short Taxonomy and Comparative Results. In *Proc. Workshop on On-Line Decision-Making in Multi-Robot Coordination, IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–8, 2015.

[7] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[8] Julian De Hoog, Stephen Cameron, and Arnoud Visser. Role-based autonomous multi-robot exploration. In *Proc. Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns*, pages 482–487, 2009.

[9] Julian De Hoog, Stephen Cameron, Arnoud Visser, et al. Autonomous multi-robot exploration in communication-limited environments. In *Proc. 11th Conference Towards Autonomous Robotic Systems*, 2010.

[10] M. Bernardine Dias and Anthony Stentz. A Free Market Architecture for Distributed Control of a Multirobot System. In *Proc. 6th International Conference on Intelligent Autonomous Systems*, volume 6, pages 115–122, 2000.

[11] Sachithra Hemachandra, Matthew R. Walter, Stefanie Tellex, and Seth Teller. Learning Spatially-Semantic Representations from Natural Language Descriptions and Scene Classification. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2623–2630, 2014.

[12] Geoffrey A. Hollinger and Sanjiv Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, 28(4):967–973, 2012.

[13] Julian De Hoog, Stephen Cameron, and Arnoud Visser. Selection of rendezvous points for multi-robot exploration in dynamic environments. In *Proc. Workshop on Agents in Realtime and Dynamic Environments International Conference on Autonomous Agents and Multi-Agent Systems*, 2010.

[14] Matteo Luperto and Francesco Amigoni. A Constructive Machine Learning Approach for Robot Exploration and Search. In *Proc. Workshop on Machine Learning in Planning and Control of Robot Motion IEEE International Conference on Intelligent Robots and Systems*, pages 8–10, 2015.

[15] Matteo Luperto, Leone D 'Emilio, and Francesco Amigoni. A Generative Spectral Model for Semantic Mapping of Buildings. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4451–4458, 2015.

[16] Matteo Luperto, Alberto Quattrini Li, and Francesco Amigoni. A system for building semantic maps of indoor environments exploiting the

concept of building typology. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8371 LNAI:504–515, 2014.

[17] Óscar Martínez Mozos. *Semantic Labeling of Places with Mobile Robots*. Phd thesis, Albert-Ludwigs-Universität Freiburg, 2008.

[18] Hoa G. Nguyen, Narek Pezeshkian, Anoop Gupta, and Nathan Farrington. Maintaining communication link for a robot operating in a hazardous environment. In *Proc. ANS 10th International Conference on Robotics and Remote Systems for Hazardous Environments*, volume 10, pages 256–263, 2004.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2012.

[20] Max Pfingsthorn, Bayu Slamet, and Arnoud Visser. A scalable hybrid multi-robot SLAM method for highly detailed maps. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5001 LNAI, pages 457–464, 2008.

[21] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3515–3522, 2012.

[22] Andrzej Pronobis, Patric Jensfelt, Kristoffer Sjöö, Hendrik Zender, Geert-Jan M Kruijff, Oscar Martinez Mozos, and Wolfram Burgard. Semantic modelling of space. *Cognitive Systems*, 8:165–221, 2010.

[23] Alberto Quattrini Li, Riccardo Cipolleschi, Michele Giusto, and Francesco Amigoni. A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings. *Autonomous Robots*, 40(4):581–597, 2016.

[24] Martijn N. Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445, 2007.

[25] Axel Rottmann, Óscar Martínez Mozos, Cyrill Stachniss, and Wolfram Burgard. Semantic place classification of indoor environments with mobile robots using boosting. In *Proc. 12th National Conference on Artificial Intelligence, and the 17th Annual Conference on Innovative Applications of Artificial Intelligence and the 17th Annual Conference on Innovative Applications of Artificial Intelligence*, pages 1306–1311, 2005.

[26] Nicholas Roy and Gregory Dudek. Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots*, 11(2):117–136, 2001.

[27] scikit-learn. 1.17. neural network models (supervised). http://scikit-learn.org/stable/modules/neural_networks_supervised.html, 2011. [Online; accessed 2-April-2017].

[28] Weihua Sheng, Qingyan Yang, Jindong Tan, and Ning Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955, 2006.

[29] Reid Simmons, David Apfelbaum, Wolfram Burgard, and Dieter Fox. Coordination for multi-robot exploration and mapping. In *Proc. AAAI National Conference on Artificial Intelligence*, 2000.

[30] Victor Spirin, Stephen Cameron, and Julian De Hoog. Time Preference for Information in Multi-agent Exploration with Limited Communication. In *Proc. Conference Towards Autonomous Robotic Systems*, pages 34–45, 2013.

[31] Victor Spirin, Julian De Hoog, Arnoud Visser, and Stephen Cameron. MRESim, a Multi-robot Exploration Simulator for the Rescue Simulation League. In *RoboCup 2014: Robot World Cup XVIII*, pages 106–117. Springer, 2015.

[32] Cyrill Stachniss, Óscar Martínez Mozos, and Wolfram Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1692–1697, 2006.

[33] Regis Vincent, Dieter Fox, Jonathan Ko, Kurt Konolige, Benson Limketkai, Benoit Morisset, Charles Ortiz, Dirk Schulz, and Benjamin Stewart. Distributed multirobot exploration, mapping, and task allocation. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):229–255, 2008.

[34] Arnoud Visser and Bayu A. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *Proc. 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops*, pages 680–687, 2008.

[35] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proc. 2nd International Conference on Autonomous Agents*, pages 47–53, 1998.

Appendix A

Further environments used

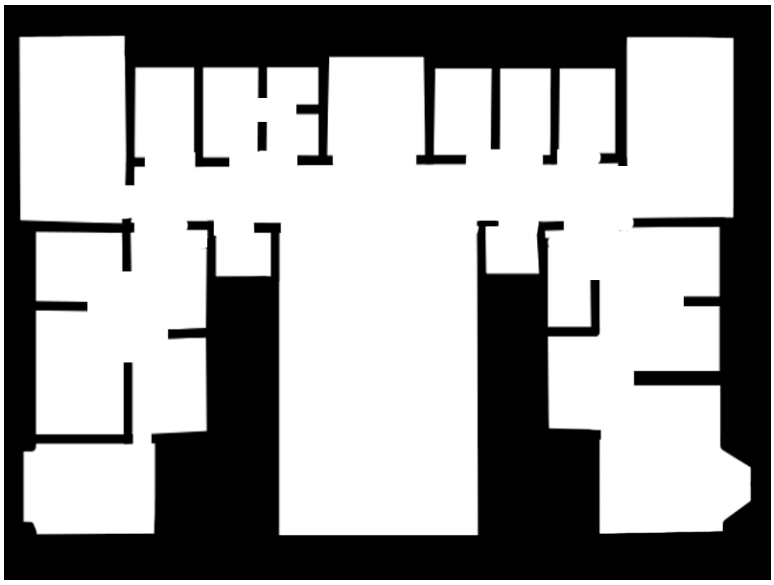


Figure A.1: Environment battlecreekhs

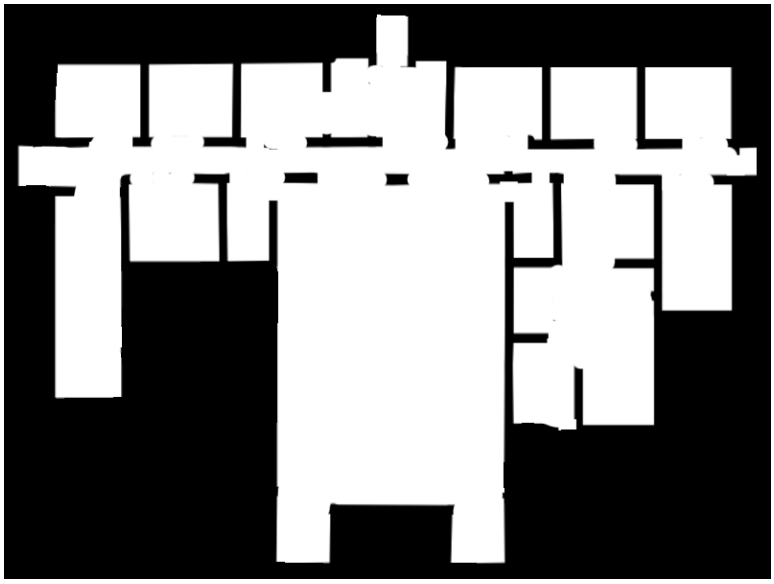


Figure A.2: Environment herndon

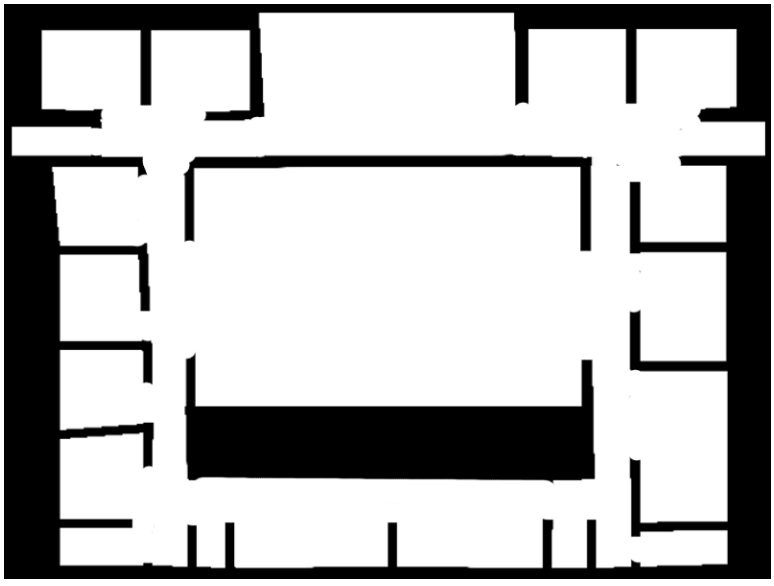


Figure A.3: Environment plans

Figure A.4: Environment rockisland

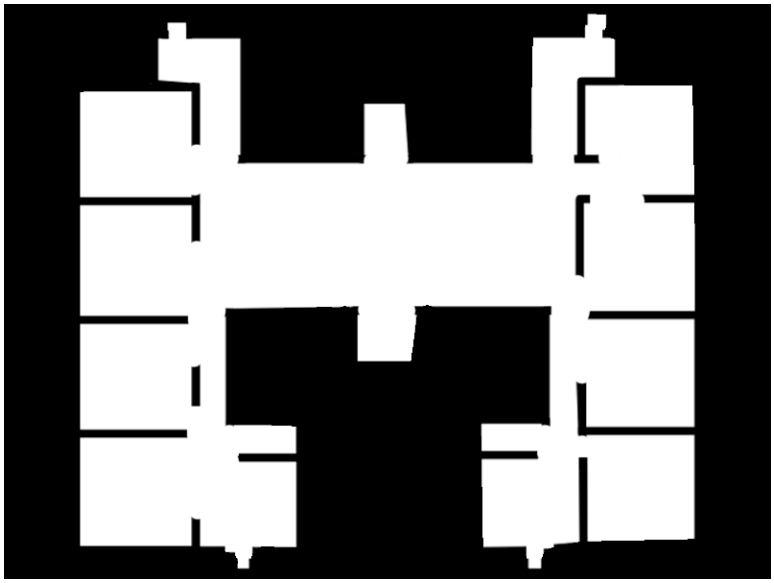


Figure A.5: Environment scuolasconosciuta

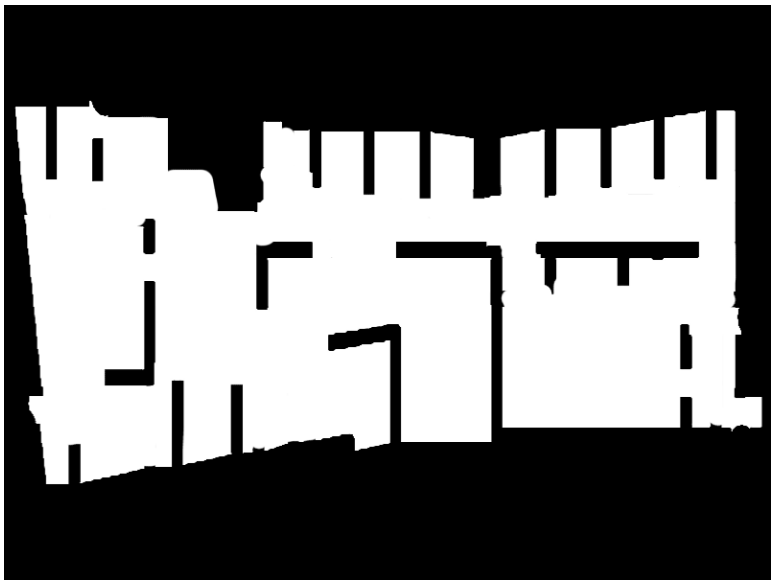


Figure A.6: Environment vipiteno