University of “Roma TRE”
Faculty of Computer Science Engineering
Computer Science and Automation department

Doctor of Philosophy Dissertation

The Localization Problem: from Robotics to Sensor Networks

by

Andrea Gasparri

Supervisor
prof. **Lorenzo Sciavicco**

Reviewer
Prof. **Giuseppe Oriolo**

Reviewer
Prof. **Gaurav S. Sukhatme**

Rome, 2007

"I thought you were dead" – Detective Del Spooner
"Technically I was never alive, but I appreciate your concern" – Sonny
. . . I, Robot

Acknowledgements

I think what is really tough when thanking people is not to leave anybody out... For this reason I want to apologize ahead to the people who, no matter how hard I try, I am going to forget...

I feel grateful and wish to thank the research group I have shared this experience with...

"Lorenzo Sciavicco, Stefano Panzieri, Federica Pascucci and Giovanni Ulivi"

I want also to thank Alessandro Longhi who makes everything easier in the Robotics laboratory with his Swiss constancy and organization...

In the same way, I do have to thank all the people who have supported me and believed in me along this path....

I am talking about my beloved parents Piero and Natalina, my irreplaceable brother Paolo, my always present aunt Antonia and the little Briciola...

I am talking about my closest friends Andrea and Paola who I have shared almost everything from moments of happiness to the most deep crises...

I am talking about my colleague and dear friend Mattia, who I have been able to profitably work with while having a lot of fun...

I am talking about my girlfriend Wendy, who has found herself forced to proof-read all this paranoia with the only fault to be an english native speaker...(I will never be able to thank her enough)...

To conclude, I am talking about this beautiful person who suddenly left me after my graduation but who I am sure still keeps watching, advising and proudly following my life....my elementary teacher Ismene...

With deep gratitude,
Andrea

Preface

The research described in this thesis concerns the Localization Problem within two different fields of interest: Mobile Robotics and Sensor Networks. Over the past decade, a strong interest was shown by the research community in building heterogeneous frameworks able to integrate robotics components with sensorial devices. Interest in developing these systems comes from the astonishing number of possible applications, which can be conceived by exploiting their inherent robustness and flexibility.

This thesis is motivated by the observation that in order to build such complex systems some basic services must be provided in advance. Among them, the localization service holds an important rule in both fields: robots must know their location while performing a task in order to safely interact with their environment, while sensor devices must be aware of their location (at least roughly) in order to properly supply services.

The thesis is organized into three parts:

- Part I addresses the localization problem in the robotics context. Initially, the state of the art is presented. Afterward, three different approaches are proposed and simulations along with experimental results are given. Finally, a comparative analysis is performed and results are discussed.
- Part II addresses the self-localization problem in sensor networks. After a preliminary analysis of the state of the art, two approaches coming from the probabilistic framework are derived, and their capabilities are investigated through simulations and experimental results. Finally, a comparative analysis is performed and results are discussed.
- Part III proposes a possible integrated framework for the coverage problem. The classic scenario is extended by a sensor network which provides coordination to the robots by exploiting its distributed nature. In particular, this work investigates how the dynamics of the coverage problem are modified by the introduction of such a sensor network. Finally, conclusions are drawn and future work is proposed.

Contents

Part I

Localization In Mobile Robotics

Chapter 1

Introduction

Mobile Robotics is a challenging interdisciplinary field. Research groups who converge to this area are often characterized by different theoretical backgrounds stemming from mechanical to artificial intelligence. Since a mobile robot has to move around within its environment and perform autonomously almost any task, it must be able to safely interact with it. In order to achieve that, a localization module should always be included in a robotics control architecture. This module should provide reliable pose information for the robot even in the presence of noisy data and unpredictable environmental interactions. Due to the difficulty of obtaining reliable pose information, the localization problem has been a highly active field of research over the last two decades.

1.1 The Localization Problem in Mobile Robotics

In mobile robotics, one of the most important goals is to realize the complete autonomy of the robot. A mobile robot must be able to safely interact with its environment in order to achieve such autonomy. For this reason, the availability of reliable pose information is critical. The localization problem aims to estimate the robot's pose in an environment, using data coming from sensors. However, the interaction between the robot and the environment, along with the presence of noisy sensors readings make the problem extremely challenging.

The localization problem is usually classified into three different problems: position tracking, global localization and kidnapped robot. This classification, which reflects the way the research community approached the problem over the years, underlies an increasing complexity due to the progressive release of some simplifying assumptions.

Initially the research community, given a starting position of the robot, has focused its attention on developing techniques for keeping track of the robot's current position by exploiting both exteroceptive (e.g. sensor readings) and interoceptive (e.g. dead-reckoning) information (i.e. position tracking problem). Successively, with the release of this assumption, new techniques were investigated focused mainly on maintaining the multi-hypothesis until some evidences were achieved (i.e. global localization problem). Finally, the problem of having new data that force the estimation of an already localized robot at a completely different position has been taken into account (i.e. kidnapped robot problem).

The emergence of Multi-Robot Systems (MRS) has gained great attention in recent years. Indeed, a team of robots brings several interesting advantages. The reliability of a multi-robot system is higher as tasks can be performed even if a member fails. In addition, the time required to execute a task is often significantly sped up. However, cooperation and collaboration underlying a multi-robot system introduce new challenges. In regard to the localization problem, algorithms originally developed for the single-robot context may be used by parallelization: an instance of the algorithm for each robot. Unfortunately, this approach does not take into consideration the inherent collaborative and cooperative nature of a multi-robot system . Conversely, a better localization accuracy can be generally obtained when collaboration among robots is taken into account. Therefore, new paradigms have been proposed to properly exploit all the information available.

1.2 State of the art

1.2.1 Single-Robot

The Localization problem has been widely investigated by several research communities in the last two decades. Historically, the position tracking, i.e. the instance of the localization problem where a prior knowledge about the starting position of the robot is available, has been the first problem to be investigated.

The probability theory has over the years proved to be a powerful framework for modeling the localization problem as a stochastic estimation problem. From a probabilistic standpoint, the state of the robot (pose), which is defined in terms of position (x, y) and orientation θ, is described by means of a probability distribution called *belief* over the whole state space.

The Kalman Filter is likely the most famous approach based on this framework [86]. Kalman based methods represent the *belief* by means of a Gaussian distribution over the state space of the robot: the mode of the distribution yields the current robot position, while the variance represents the accuracy of the estimation. Gaussian distribution, described by means of only two parameters, has two important advantages: from a mathematical point of view a discretization of the state space is not required [107], while from a computation standpoint, an on line implementation can be easily faced [121].

Some works for relative localization based on dead-reckoning exploit Kalman Filter techniques to improve the accuracy of their estimations. [16, 134]. Dead-reckoning is a popular technique used to estimate the current position of a robot by exploiting simple geometric equations (the kinematics of the robot) on odometric data. Since the starting position of the robot is assumed as its reference frame, this technique provides a relative localization of the robot. In particular, in [16] a low-cost solid-state inertial navigation system (INS) for mobile robotics applications is described. Error models for the inertial sensors are generated and included in an extended Kalman filter (EKF) for estimating the position and orientation of a moving robot vehicle. However, this approach is limited by the fact that orientation information are not exploited for the calculation of the position. In [134] the mobile robot localization problem is decomposed into two stages; attitude estimation followed by position estimation. Two Kalman filters are exploited to form the ”smoother” in the attitude estimation loop. The smoother exploits the special nature of the data fused: high frequency inertial sensor (gyroscope) data and low frequency absolute orientation data (from a compass or sun sensor). During each time interval one of them propagates the attitude estimate forward in time until it is updated by an absolute orientation sensor. At this time, the second filter propagates the recently renewed estimate back in time. This way, the limited observability of the system is optimally exploited by combining the outcome of the two filters.

Other works use Kalman filter techniques to perform absolute localization exploiting beacons, landmarks or satellite signals [100, 20]. In [100] the authors developed a system in which the basic localization algorithm is formalized as a vehicle-tracking problem, employing an EKF to match ”geometric” beacon observations (environment features) to a navigation map to maintain an estimate of the mobile robot. Geometric beacons, introduced in [60, 61], are a special type of targets with two important properties that can be reliably observed in successive sensor measurements and that can be accurately described by means of a concise geometric parametrization. In [20], the authors proposed a theoretical development and experimental implementation of a complete navigation procedure for use in an autonomous mobile robot for structured environments. Estimates of the vehicle’s position and orientation are based on the rapid observation of visual cues located at discrete positions within the environment. In this context, the extended Kalman filter is used to combine these visual observations with sensed wheel rotations to produce optimal estimates continuously. The complete estimation procedure as well as the control algorithm developed are time independent.

Finally, some approaches use Kalman Filter techniques to combine both relative and absolute sensor data [73, 121]. In [73] a low-cost strategy based on well calibrated odometry is presented for localizing mobile robots. The paper describes a two-step process for correction of “systematic errors” in encoder measurements followed by fusion of the calibrated odometry with a gyroscope and GPS resulting in a robust localization scheme. A Kalman filter operating on data from the sensors is used for estimating position and orientation of the robot. In [121] the authors propose a localization algorithm which exploits a Kalman Filter for the integration of data coming from both interoceptive and exteroceptive sensors such as encoders, gps, inertial platform and laser range-finders. The localization system is integrated within a navigation system providing a reliable feedback to it.

Kalman based approaches have proven over the years to be an effective and robust solution for tracking the robot pose. However, Gaussian distributions cannot be applied to describe the multi-hypothesis. Therefore, these methods are ineffective to deal with the global localization problem, i.e. the instance of the localization problem where no prior knowledge of the robot starting position is available.

Some works provide extensions of methodologies introduced to solve the position tracking problem [10, 85, 7]. The underlying idea of these approaches is the concept of parallelization: allocating a (Gaussian) probability distribution for each plausible hypothesis and then providing a technique for selecting the most likely hypothesis over time with respect to sensor evidence. In [10] several Gaussian hypotheses are exploited to represent the probability distribution of the robots location in the environment. In [85] the authors propose a hybrid localization method which exploits multiple Kalman Filters for hypothesis tracking and probability theory for evidence fusion. A feature-based description of the environment is required for the hypothesis generation. In [7] hypotheses are generated using a constraint-based search in the interpretation tree of possible local-to-global-pairings. This way, a set of continuously located position hypotheses of unbounded accuracy is obtained. The same approach holds for tracking: hypotheses are tracked and split as soon as location ambiguities arise from uncertainties and sensing.

Unlike the approaches described thus far, which are characterized by a continuous description of the probability distribution, other works introduce suitable discretization of the state space.

Some approaches are based on the construction of certainty grid maps [37, 36]. Certainty grid maps, introduced by [114], have been initially designed to provide a probabilistic model of the robots environment. They have also been successfully used for collision avoidance [25, 26, 27] and path planning [33, 113]. In [37], grid maps have been exploited for the estimation of the absolute position of a robot. The idea is to accumulate in each cell of the position probability grid the posterior probability of this cell referring to the current position of the robot. As the pose of a robot is described entirely by its position and orientation, the discretization of the state space leads to a three dimensional problem. Although this approach provides interesting results, it suffers from excessive computational overhead [36].

Other works rely on Monte Carlo Integration methods[59] for the discretization of the state space [67, 97, 162]. These methods were first investigated in the early 70's [77, 76, 3]. However, they have been neglected for almost two decades due to the lack of computational capability. They have been rediscovered around the 90's as a consequence of the significant technological progresses.

Nowadays, Monte Carlo techniques are successfully applied to solve estimation problems in several research areas, such as computer vision [83], wireless telecommunications [152] and chemistry [55]. In [67] the authors propose a Monte Carlo based algorithm called Monte Carlo Localization (MCL) for mobile robot localization. This approach represents an evolution of the earlier works based on grid maps. In fact, although the probabilistic framework has been maintained, a sampling-based method for approximating the probability distribution is introduced. This way, the computational complexity is lowered. In [97] the authors introduce Real-Time Particle Filters (RTPF) to deal with constraints imposed by limited computational resources. This approach make use of all sensor information even when the filter update rate is below the update rate of the sensors. This is achieved by representing posteriors as mixtures of sample sets, where each mixture component integrates one observation that arrives during a filter update. The weight of the mixture components are computed so as to minimize the approximation error introduced by the mixture representation. This way, the computational resources are focused only on valuable sensor information. In [162] the authors propose an improvement of the previously described RTPF to make more efficient the trade-off between accuracy, simplification and robustness. This is achieved by exploiting probability retracing, consecutive window filtering, and adaptive particle set size.

Although Monte Carlo Integration methods have the significant advantage of approximating (almost) any distribution they suffer from the degeneracy (aka depletion) problem, i.e., the problem of having most of the particles with a negligible weight after few iterations [8, 58]. This weakness is problematic especially to deal with the kidnapped robot problem, i.e. the problem of having new data that force the estimation of an already localized robot at a completely different position.

Different techniques have been proposed to deal with the depletion of samples [103, 160, 110, 98]. This phenomenon might be naively mitigated increasing the number of particles used to approximate the distribution. Unfortunately, in this way an increasing computational effort is experienced. A better way to approach the degeneracy phenomenon is to introduce a resampling step in order to augment the diversity among particles [92]. In addition, a proper measure of such a degeneracy might be introduced so that the resampling step is performed anytime a significant depletion of the particles set is observed [103]. In [57], to further mitigate this phenomenon, the authors suggest a suitable candidate of the important function, i.e., a candidate that minimizes the variance of the importance weights conditioned upon the data [160]. In [110], a cluster-based extension of the MCL method is proposed. This algorithm introduces the idea of clusters of particles to track multiple distinct hypotheses, where each cluster is considered as an independent hypothesis about the robot's pose. It works on two different levels: at particle level, the classical Bayesian formulation is adopted to update hypotheses, while at cluster level, the hypothesis with the highest probability is used to determine the robot's pose. Although distinct hypotheses can be maintained over time, having a constant number of clusters makes the kidnapped robot problem more difficult. In fact, as robots start moving, the coverage of the environment is no longer guaranteed.

In [98] the authors propose an alternative re-sampling schema, based on genetic algorithms, to mitigate the sample impoverishment problem. The algorithm is able to maintain the diversity of particles during the resampling process by means of the crossover genetic operator. However, it has been conceived only to solve the position tracking problem on a landmark-based framework.

1.2.2 Multi-Robot

The emergence of Multi-Robot Systems (MRS) introduces new challenges for the localization problem. In fact, the inherent collaborative and cooperative nature of these systems requires new paradigms to be properly exploited. Indeed, frameworks for solving the localization problem in the multi-robot context might be naively obtained by extending classical approaches developed for the single robot context, e.g parallelizing their execution. However, this way the inherent collaborative nature of the system is completely neglected. Instead, better results can be obtained by taking into account all the available information.

Multi robot systems can be classified in regards to their architecture into two categories: *centralized* and *decentralized* [41]. Centralized architectures are characterized by a single control robot (leader) that is in charge of organizing the activities of the other robots. The leader takes part in the decision process for the whole team, while the other members act according to the dispositions of the leader. Conversely, decentralized architectures are characterized by "self-organization", i.e., each robot is autonomous in the decision process with respect to each other. However, all robots share a common goal and their actions are toward its achievement.

Localization techniques have been developed with respect to these two architectures. In a centralized system, a leader collects data provided by the team and performs the localization process for the whole group. In a decentralized system, each robot performs its estimation and exchanges data with the other robots to improve the localization process. Both paradigms present advantages as well as drawbacks. Normally, the assignment of a task is easier in a centralized system compared to a distributed one, as the leader is the only one in charge of it. Additionally, centralizing the computation requires only one robot, or few if redundancy is taken into account, with suitable hardware capabilities. However, this leads to a lack of robustness as, once a leader fails, the system becomes unable to accomplish the task. These disadvantages can be overcome removing the central processing unit and spreading all the decisional issues over the team. This way, since each robot acts independently, modularity and robustness are achieved

[122]. Obviously, suitable hardware capabilities for all robots are required in this case.

Moreover, in a centralized fashion, a supervisor collects all the data coming from the robots and provides an estimate of the poses for the robots in the team. This approach requires all members to continuously communicate with the supervisor. In order to maintain the communication, robots need either to move closely to the supervisor or to implement a mobile ad-hoc network. Therefore some constraints on robots mobility have to be defined to guarantee at least one communication-path from any robot to the supervisor at each time instant. The decentralized approach, referred in literature as *collaborative or cooperative* localization, assumes that each robot in the workspace uses its own sensors, exchanges data only with other robots within its neighborhood, and runs a local algorithm to estimate its own pose.

In [96] the concept of *mobile landmark* is introduced. The authors consider a team of robots exploring an unknown environment without any beacon. The exploration is carried out using the robots themselves as landmarks. Each vehicle repeats move-and-stop actions and acts as a landmark for the other robots, while a data fusion algorithm collects data to improve the estimate of the relative positioning of the robots. According to the authors, this mechanism works well in uncharted environments since the concept of landmark is intrinsically exploited. In [127], the idea previously introduced is exploited to improve the exploration of an unknown environment. In detail, underlining how the odometry errors might heavily affect the mapping of the environment, the authors introduce a mapping technique which acts also to minimize the effects of inherent navigation. A similar solution is proposed in [129, 130] where a new sensing strategy, named *robot tracker*, is exploited to improve the accuracy of the pose estimation of each robot. The robots explore the environment in teams of two; each platform is equipped with a robot tracker sensor that reports the relative position of the other robot. Measurements are used in a particle filter to update the poses of the multi-robot system together with the associated uncertainties. All the solutions above mentioned suffer from the following limitations: only one robot is allowed to move at any given time, and the team has to maintain sensorial contact at all times.

A different collaborative scheme, based on estimation theoretical framework, is presented in [65], where two robots are supposed to navigate in a partially known environment. At every meeting they stop and improve their localization by exchanging their *beliefs*, i.e. the posterior probability density over the state space conditioned to measurements. A particle filter is at the base of the algorithm, giving the possibility to handle a non Gaussian shaped belief, and achieve localization. Another promising solution is proposed in [133, 115] and reviewed in [105, 106], where a Kalman based algorithm is used to realize collaborative localization. During the navigation cycle, each robot collects data from its proprioceptive sensors to perform the prediction step of a Kalman filter while sharing information from the exteroceptive sensors with the rest of the team during the update. The authors introduce a distributed algorithm based on singular value decomposition of the covariance matrix. In this way, the centralized filter is decomposed into a number of smaller communicating filters, one for each robot. However, this approach can be applied only if inter-robot communication can be consistently guaranteed. If not, problems related to the maintenance of cross-correlations terms arise. In [82], a distributed approach based on maximum likelihood estimation is described. Robots are equipped with sensors that allow them to measure the relative pose and identity of nearby robots, as well as sensors that allow them to measure changes in their own pose. Therefore, localization is obtained using only the robots themselves as landmarks. In [135], the authors focus on the problem of examining the effect on localization accuracy of the number N of participating robots and the accuracy of the sensors employed. In detail, the improvement in localization accuracy per additional robot as the size of the team increases is investigated.

1.3 Part I Organization

The first part of the dissertation is focused on the Localization problem in Mobile Robotics. The remaining portion of Part I is organized as follows. In Chapter 2 an improved Monte Carlo Filter is proposed along with some experimental results. In Chapter 3 a spatially structured evolutionary approach is presented and experimental results are discussed. In Chapter 4 an alternative biology-inspired framework is devised and experimental results are given. In Chapter 5 these algorithms are compared in terms of performance and computational complexity and conclusions are drawn. Note that, in order to make the treatment of each subject as self-contained as possible, each chapter is provided with a preliminary theoretical background, the description of the problem settings and the discussion of individual simulations and experimental results.

Chapter 2

Monte Carlo Clustered Evolutionary Filter

In this chapter an enhanced Monte Carlo Filter is proposed: the Clustered Evolutionary Monte Carlo Filter (CE-MCL). This algorithm attempts to overcome the classical Monte Carlo Filter drawbacks such as the depletion problem. In order to achieve that, it takes advantage of an evolutionary approach along with a clustering method. In particular, the former is exploited to quickly find out local maxima, whereas the latter, being dynamical, helps to obtain an effective exploration of the environment. The ability to provide a smart partition set of the research space along with the guarantee to converge within each subset, make the algorithm able to solve the localization problem and maintain the multi-hypotheses. Exhaustive experiments, carried on the robot ATRV-Jr manufactured by iRobot, are shown to prove the effectiveness of the proposed CE-MCL filter.

2.1 Monte Carlo Integrations Methods

A suitable framework for the localization problem can be devised exploiting the probability theory. From a probabilistic point of view, the robot's pose can be described by a probability distribution called *belief*. As a result, the localization problem consists of estimating the *belief* over the state space conditioned on the data.

A Bayesian framework to estimate this probability distribution, called *Markovian Framework*, has been introduced in [66]. The key idea is to recursively compute the *belief* by means of the Bayes rule as new sensors measurement comes. In literature the *belief* is defined as:

$$Bel(x_k) = p(x_k \mid D_k),\ x \in \Xi, \tag{2.1}$$

It represents the probability to have the robot at location x_k at time k, given all the data D_k up the time, where Ξ is the set of all poses.

In mobile robotics, data (D_k) can be broken down into *control data* (U_k)and *perceptual data* (Z_k). Control data represents the inputs of the system and, as they are not always known, are retrieved by encoders or other proprioceptive sensors. Perceptual data represents information about the environment, such as laser measurements. As a consequence, *prior* and *posterior belief* can be defined as follows:

$$Bel^-(x_k) = p(x_k \mid U_{k-1}, Z_{k-1}). \tag{2.2}$$

that it is the belief the robot has got, after the integration of the control data u_{k-1}, and before it receives the perceptual data z_k.

$$Bel^+(x_k) = p(x_k \mid U_{k-1}, Z_k). \tag{2.3}$$

that is the belief the robot has got once the perceptual data z_k has been integrated. Regarding to the integration data, several considerations need to be made:

i) Using the *Total Probability Theorem* the $Bel^-(x_k)$ can be rewritten as:

$$\begin{aligned} Bel^-(x_k) = \int_\Xi & p(x_k \mid x_{k-1}, U_{k-1}, Z_{k-1}) \times \\ & p(x_{k-1} \mid U_{k-1}, Z_{k-1}) dx_{k-1}. \end{aligned} \tag{2.4}$$

The equation states that the prior belief of being in state x_k is the sum of the probabilities of coming from state x_{k-1} to state x_k given all the earlier sensor measurements. The second term of the integral represents the *belief* at time $(k-1)$, as the robot pose at generic step k does not depend on the action that is performed at the same step. To further simplify the formulation, the assumption to have a *Markov* environment can be introduced. The key idea is to consider the past and future data independent, with the knowledge of the current state [66]. As a consequence the prior belief can be written as:

$$\begin{aligned} & Bel^-(x_k) = \int_\Xi p(x_k \mid x_{k-1}, u_{k-1}) \times \\ & Bel^+(x_{k-1}) dx_{k-1}. \end{aligned} \tag{2.5}$$

ii) Using the *Bayes rule* the *posterior* can be rewritten as:

$$\begin{aligned} Bel^+(x_k) = p(z_k \mid x_k, U_{k-1}, Z_{k-1}) \times & \\ \frac{p(x_k \mid U_{k-1}, Z_{k-1})}{p(z_k \mid U_{k-1}, Z_{k-1})} & \end{aligned} \tag{2.6}$$

The equation states that the posterior belief is the conditional probability of observing z_k, weighted by the ratio of the prior belief of being in state x_k, $Bel^-(x_k)$, and the probability of observing measurement z_k conditioned on all information so far. To further simplify the formulation, the *Markov* assumption can be adopted again. As a result the posterior belief can be rewritten as:

$$Bel^+(x_k) = \frac{p(z_k \mid x_k)\ Bel^-(x_k)}{p(z_k \mid U_{k-1}, Z_{k-1})} \tag{2.7}$$

iii) As a combination of the equations mentioned above, the recursive formulation for localization is:

$$\begin{aligned} Bel^{+}(x_k) = \eta\, p(z_k \mid x_k) \times \\ \int_{\Xi} p(x_k \mid x_{k-1}, u_{k-1}) Bel^{+}(x_{k-1}) dx_{k-1}, \end{aligned} \tag{2.8}$$

where η represents $p(z_k \mid U_{k-1}, Z_{k-1})$ and can be viewed as a normalization factor.

As the integrals above are not tractable, several efforts have been devoted to approximate the state space in order to make the recursive equation above simple to be computed.

Monte Carlo integrations methods extend the Markovian framework by means of a sampling approach to represent the posterior distribution (belief). These methods have the significant advantage of not being subject to any linearity or Gaussianity constraints on the model, and they also offer interesting convergence properties. As a consequence, these methods turn out to be a powerful tool to deal with the global localization problem. The *Perfect Monte Carlo Sampling* draws N independent and identically distributed random samples $\{x_k^{(1)}, \ldots, x_k^{(N)}\}$ according to $Bel^{+}(x_k)$. Consequently, the approximation of the posterior distribution is given by

$$Bel^{+}(x_k) \approx \frac{1}{N} \sum_{i=1}^{N} \delta_{x_k^{(i)}}(x_k - x_k^{(i)}), \tag{2.9}$$

where $\delta_{x_k^{(i)}}(x_k - x_k^{(i)})$ represents the delta-Dirac mass located in $x_k^{(i)}$. However, due to the difficulty of efficiently sampling from the posterior distribution $Bel^{+}(x_k)$ at any sample-time k, a different approach is required. An alternative solution is the *Sequential Importance Sampling* approach. The key idea is of drawing samples from a normalized *importance sampling distribution* $\pi(x_k \mid d_k)$ which has a support including that of the posterior belief $Bel^{+}(x_k)$. In this case, the approximation of the posterior is given by

$$Bel^{+}(x_k) \approx \sum_{i=1}^{N} w_k^{(i)} \delta_{x_k^{(i)}}(x_k - x_k^{(i)}), \tag{2.10}$$

where the *importance weight* is computed as

$$w_k^{(i)} = w_{k-1}^{(i)} \cdot \frac{p(z_k \mid x_k^{(i)}) Bel^{-}(x_k)}{\pi(x_k \mid d_k)}. \tag{2.11}$$

In mobile robotics, a suitable choice of the importance sampling distribution $\pi(x_k \mid d_k)$ is the prior distribution $Bel^{-}(x_k)$ [78]. With this choice, the importance weight can be easily computed as:

$$w_k^{(i)} = w_{k-1}^{(i)} \cdot p(z_k \mid x_k^{(i)}), \tag{2.12}$$

and the importance sampling distribution can be written in a recursive fashion:

$$\pi(x_k \mid d_k) = p(x_k \mid x_{k-1}, u_{k-1}) \cdot Bel^+(x_{k-1}). \tag{2.13}$$

Such formulation has the advantage of allowing an on-line evaluation of the importance weight as long as new data is available; however it causes the degeneracy problem, i.e. the problem of having most of the samples with a negligible weight after few iterations. A common approach to overcome this problem is to provide a resampling step, which aims to replace particles with small importance weight by means of a suitable strategy. The algorithm 1 shows a typical implementation schema for a *Sequential Monte Carlo filter with resampling step.* The majority of works in literature relies on this schema, with a specialization for the resampling approach adopted.

Algorithm 1: Sequential Monte Carlo Filter

```
Data: $Bel^+(x_{k-1}) = \{x_{k-1}^{(i)}, w_{k-1}^{(i)}\}$ , $u_{k-1}$ , $z_k$
Result: $Bel^+(x_k)$

/* Importance Sampling                                                  */
1  Compute $\pi(x_k \mid d_{k-1}) = p(x_k \mid x_{k-1}, u_{k-1}) \cdot Bel^+(x_{k-1})$
2  for $i$=1 to $Ns$ do
3      Sample $\tilde{x}_k^{(i)} \sim \pi(x_k \mid d_{k-1})$
4      Evaluate $w_k^{*(i)} = w_{k-1}^{*(i)} \cdot \frac{p(z_k|\tilde{x}_k^{(i)})Bel^-(x_k)}{\pi(x_k|d_{k-1})}$
5  end
/* Normalization                                                        */
6  for $i$=1 to $Ns$ do $\tilde{w}_k^{(i)} = \frac{w_k^{*(i)}}{\sum_{j=1}^{Ns} w_k^{*(i)}}$
7  Evaluate $N_{eff} = \frac{1}{\sum_{i=1}^{Ns}(\tilde{w}_k^{(i)})^2}$
/* Degeneracy Test                                                      */
8  if $N_{eff} \geq N_{thres}$ then
9      $\{x_k^{(i)}, w_k^{(i)}\} = \{\tilde{x}_k^{(i)}, \tilde{w}_k^{(i)}\}$
10 else
       /* Resampling                                                    */
11     $\{x_k^{(i)}, w_k^{(i)}\} = ResamplingProcedure(\{\tilde{x}_k^{(i)}$ , $\tilde{w}_k^{(i)}\})$
12 end
13 $Bel^+(x_k) = \{x_k^{(i)}, w_k^{(i)}\}$
```

2.2 The Clustered Evolutionary Monte Carlo Filter

The *Clustered Evolutionary Monte Carlo* filter (CE-MCL) has been conceived following the classical Sequential Monte Carlo filter schema mentioned above. The algorithm works on two different levels:

- Local level.
- Global level.

At local level the algorithm finds out local maxima within each cluster, whereas at global level the best hypothesis is obtained by a comparison among the optimal solutions provided by each cluster.

In order to realize such behavior, the algorithm introduces two strategies:

- A dynamical clustering at resampling step.
- An evolutionary action at each time-step.

2.2.1 Dynamical Clustering

The dynamical clustering provides a collection of particles subset that represents the best partition of the environment and where the probability to find out the real robot location is higher. Cluster identification is performed by means of the DBscan algorithm, which relies on a density-based notion of clusters [64]. Such algorithm offers several good properties, such as the ability of finding out clusters of arbitrary shapes, the advantage of collecting the *noisy* points, and an acceptable computational complexity. In particular, the possibility of collecting all the points belonging to any cluster turns out to be very useful in this context. In fact, it can be viewed as another mean to improve the diversity among particles. Moreover, in order to guarantee both a minimal coverage of the environment and further mitigate the degeneracy problem, a random action is introduced along with the dynamical clustering at resampling step. Such action reduces the similarity among particles randomly drawing a percentage of new samples.

2.2.2 Evolutionary Action

The evolutionary action, instead, is introduced to quickly find out local maxima within each cluster. From a genetic point of view a cluster represents the population, while the state space vector is the encoding string, e.g. the chromosome. The model of the sensor $p(z_k \mid x_k)$ is adopted as fitness function. This choice makes the local maxima to be prominent candidates to localize the robot, being the $p(z_k \mid x_k)$ part of the importance weight formulation as well. The evolutionary action is based on the probabilistic operators:

- Mutation.
- Crossover.

Mutation creates a fixed percentage of new particles sampling from a circular area centered on the selected chromosome (Fig. 2.1), whose radius is defined as follows:

$$\rho_k = \frac{1}{\sqrt{w_k^{(i)}}}. \tag{2.14}$$

The idea of defining the radius as an inverse function of the importance weight,

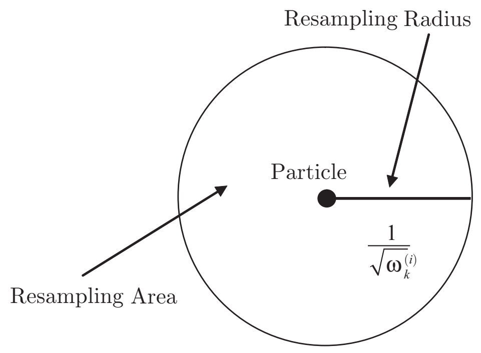

Figure 2.1: Choice of resampling area for mutation

reflects the consideration that particles, with a considerable importance weight should be located closer to the real robot than the others. Therefore, filling this area should be promising for a quicker localization. On the other hand, crossover creates a fixed percentage of new particles combining chromosomes belonging to the same cluster. The idea of selecting parents within the same subset, avoids unlikely recombination to happen, being clusters spatially organized.

At the end, the Clustered Evolutionary Monte Carlo Filter, taking advantage of these strategies, is able to both globally localize the real robot location and solve the kidnapped robot problem, as well as to maintain the multi-hypotheses.

The algorithm 2 shows a possible implementation schema for *The Clustered Evolutionary Monte Carlo Filter*.

2.2.3 Computational Complexity

In order to evaluate the computational complexity of the algorithm, several analyses have been performed. A detailed theoretical study has been done along with an empirical validation of the obtained results. According to such a study, two different cases have to be taken into account:

- *Simple* step.
- Resampling occurrence.

In the first case, when the resampling is not considered, the complexity of the algorithm turns out to be $\mathcal{O}(N \cdot M)$, where N is the number of particles and M is the number of segments describing the environment. Conversely, when the resampling occurs, the DBscan effort has to be considered. In this case the overall complexity of the algorithm is given by $max\{\mathcal{O}(N \cdot log(N)), \mathcal{O}(N \cdot M)\}$, where $\mathcal{O}(N \cdot log(N))$ is the computational load of the DBscan [64]. Two remarks are now in order: the first one is that the number of segments (M) is usually at least one order of magnitude smaller than the number of particles (N); the second is that the resampling step, during a typical execution, takes less than

Algorithm 2: Clustered Evolutionary Monte Carlo Filter

```
   Data: $Bel^+(x_{k-1}) = \bigcup_{j=1}^{N_{clust}} \{x_{k-1}^{(i)}, w_{k-1}^{(i)}\}_j$ , $u_{k-1}$ , $z_k$
   Result: $Bel^+(x_k)$
   /* Importance Sampling                                   */
1  Compute $\pi(x_k \mid d_{k-1}) = p(x_k \mid x_{k-1}, u_{k-1}) \cdot Bel^+(x_{k-1})$
2  for $i$=1 to $Ns$ do
3      Sample $\tilde{x}_k^{(i)} \sim \pi(x_k \mid d_{k-1})$
4      Evaluate $w_k^{*(i)} = w_{k-1}^{*(i)} \cdot \frac{p(z_k|\tilde{x}_k^{(i)}) Bel^-(x_k)}{\pi(x_k|d_{k-1})}$
5  end
   /* Normalization                                         */
6  for $i$=1 to $Ns$ do $\tilde{w}_k^{(i)} = \frac{w_k^{*(i)}}{\sum_{j=1}^{Ns} w_k^{*(i)}}$
   /* Evolutionary Action                                   */
7  for $j$=1 to $N_{clust}$ do
8      $\{\bar{x}_k^{(i)}, \bar{w}_k^{(i)}\}_j \leftarrow \begin{cases} Mutation(\{\tilde{x}_k^{(i)}, \tilde{w}_k^{(i)}\}_j) \\ Crossover(\{\tilde{x}_k^{(i)}, \tilde{w}_k^{(i)}\}_j) \end{cases}$
9  end
10 Evaluate $N_{eff} = \frac{1}{\sum_{i=1}^{Ns} (\bar{w}_k^{(i)})^2}$
   /* Degeneracy Test                                       */
11 if $N_{eff} \geq N_{thres}$ then
12     $[\{x_k^{(i)}, w_k^{(i)}\}_j, N_{clust}] = [\{\bar{x}_k^{(i)}, \bar{w}_k^{(i)}\}_j, N_{clust}]$
13 else
       /* Resampling                                        */
       /* 1° Step: Random action                            */
14     $\{\hat{x}_k^{(i)}, \hat{w}_k^{(i)}\} \leftarrow Random(\{\bar{x}_k^{(i)}, \bar{w}_k^{(i)}\})$
       /* 2° Step: Re-Clustering                            */
15     $[\{x_k^{(i)}, w_k^{(i)}\}_j, N_{clust}] \leftarrow DBScan(\{\hat{x}_k^{(i)}, \hat{w}_k^{(i)}\})$
16 end
17 $Bel^+(x_k) = \bigcup_{j=1}^{N_{clust}} \{x_k^{(i)}, w_k^{(i)}\}_j$;
```

10% of the overall number of iterations. Therefore, it is correct to state that the *real* complexity that should be considered is the one of the simple step: $\mathcal{O}(N \cdot M)$.

2.3 Performance Evaluation

The proposed algorithm has been tested in both simulated context and with real data in order to validate its capability to deal with the global localization problem. Several aspects have been thoroughly investigated. Particular attention has been paid to give evidence of the ability to carry on multi-hypotheses as well as to prove the ability to re-localize the robot when a kidnap occurs. Two different types of analysis have been performed. The first one has demonstrated the global algorithm aptitude to localize the robot as well as to carry on multi-hypotheses. The second more specialized analysis has proved the local algorithm capability to converge within each cluster. In the following both simulations and experimental results are reported.

2.3.1 Problem Settings

Computer Simulations Simulations have been carried out in a framework developed under Matlab by the authors. This framework provides different kinematic models for the robot, such as the unicycle model, as well as an emulation for several sensors such as a laser rangefinder. Moreover, the environment is described by a set $\mathcal{M}$ of segments. This framework supports both a complete simulated context as well a test-bed to run data coming from a real robot. These two different operative modalities turn out to be very useful, both to test the correctness and the effectiveness of the algorithm.

Real Robot Context Experiments have been carried out on the mobile platform ATRV-Jr (All Terrain Robot Vehicle Junior) manufactured by iRobot. It is a skid steering vehicle mainly designed to operate in outdoor environments. The ATRV-Jr has 4 wheels differentially driven by 2 DC motors: the motion is achieved by a differential thrust on the wheel pairs at the opposite sides. The mobile robot is equipped with 17 sonar rangefinders, a laser scanner (Sick LMS-220), an inertial platform (Crossbow DMU-6X), and a GPS receiver (Garmin GPS35-HVS). The sensory system is connected to the ATRV-Jr's on board PC (Pentium II, 350 MHz) running Linux, through serial port on a Rockeport multiserial port card. The robot is delivered with a software development environment called MOBILITY, which provides full access to the software servers available on the mobile platform. In particular, each server is assigned to control a specific hardware component (sensors and actuators). In this way all of them are reachable from the network exploiting a CORBA interface.

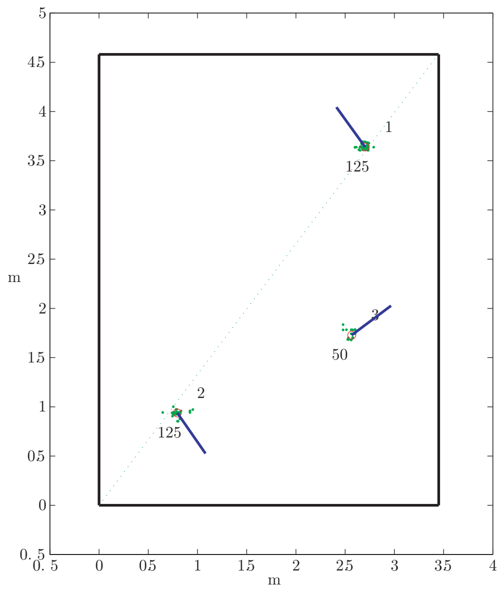

Figure 2.2: Rectangular environment: CE-MCL iteration

2.3.2 Evaluation Criteria

Two indexes of quality have been chosen to evaluate the correctness of the algorithm: the percentage of estimation failures and the entropy measurement of the particle set. The first one gives information about the reliability and the accuracy of the solution, the second one, coming from the *information theory* [140], provides a measurement of the uncertainty for a given random, and it is defined as:

$$H(\chi) = \sum_{i=1}^{n} p_i \, log_2(\frac{1}{p_i})$$

where, p_i is the probability of the i-*th* outcome for a given event χ. In this context, entropy can be properly applied to give an evaluation of the persistence of the diversity among particles.

2.3.3 Simulations

Simulations were performed to investigate the algorithm capability to carry on the multi-hypothesis. Fig. 2.2 shows a typical CE-MCL iteration step for the rectangular environment. Points (green) represent particles, whereas circles (red) are located at the center of the mass of each cluster and, segments (blue) describe the mean orientation for all particles within each cluster. Due to the environmental symmetries, at each time-step at least two subset of hypotheses are maintained, in particular the ones located at the extreme of a segment splitting the environment in two equal parts. Further, such behavior seems to be reasonable as sensor data support both locations, nothing that the laser beam range is limited to 8 m.

2.3.4 Robot in Real Environments

The robot was put into three different indoor office-like environments:

- Entire building floor.
- T-shaped hall.
- Corridor.

The first and second environments, shown respectively in Fig. 2.3 and Fig. 2.5 have been exploited to test the algorithm capability to solve both the global localization problem and the kidnapped robot problem. On the other hand, the third environment, shown in Fig. 2.7, has been exploited to investigate the local algorithm capability to converge within each cluster.

Entire building floor

Fig. 2.3 shows a typical CE-MCL estimation result for the complex environment previously mentioned. Points (red) represent the most likely hypotheses at each time step, whereas the line (blue) represents the real robot path. In particular, S is the robot start point, K is the location at which the robot is kidnapped, R is the start point after the kidnap and finally, G represents the goal point of the robot. The algorithm has been able to find out a rough estimation of the robot

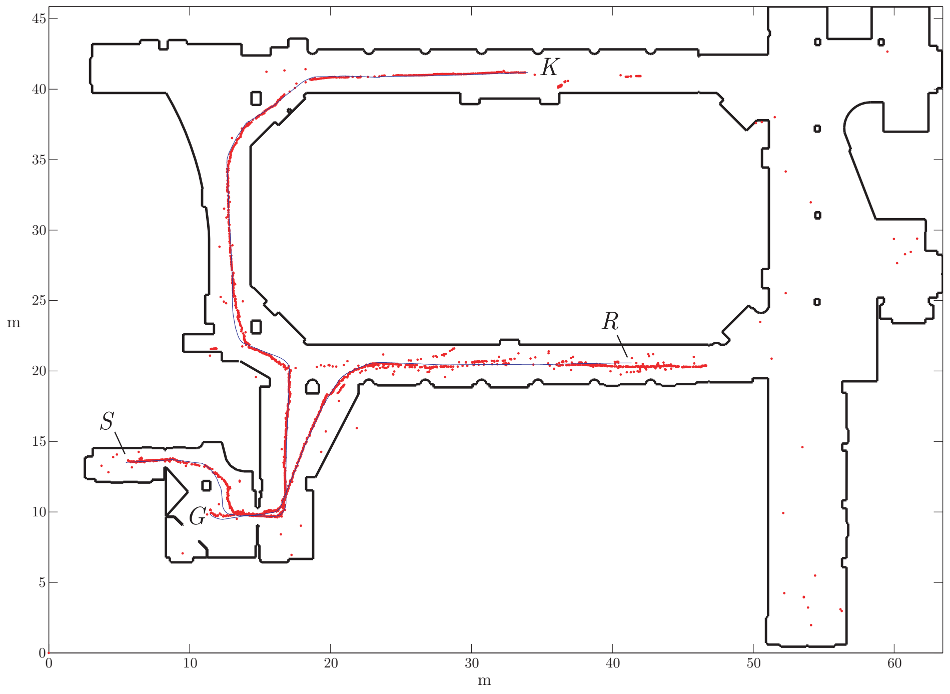

Figure 2.3: Complex environment: CE-MCL estimation result

path without any knowledge about the starting robot location. Moreover, Fig.

2.3 evidences the algorithm's ability to realize when a kidnap occurs, thus re-localize the robot. The remaining noisy points, consequence of a temporary bad estimation, prove the algorithm's tendency to explore the whole environment as well as to carry on the multi-hypotheses at each time step. Further, they might be easily removed, for instance relying on the kinematic model constraints of the mobile robot. The algorithm has been run several times in this environment to estimate the percentage of failures; the mean value settles around 30%, while the variance is ±4%. Fig. 2.4 shows the measurement of entropy for the experiment mentioned above. The red line represents the theoretical maximum entropy for the given set of particles, whereas the blue line describes the entropy during the algorithm execution, and the black line is its mean value. In order to maintain the diversity among the particle set, such value should be high. However, the algorithm should also be able to converge to the real robot location. For the CE-MCL algorithm, the mean value settles around an intermediate value, giving evidence of the algorithm's aptitude to balance both needs.

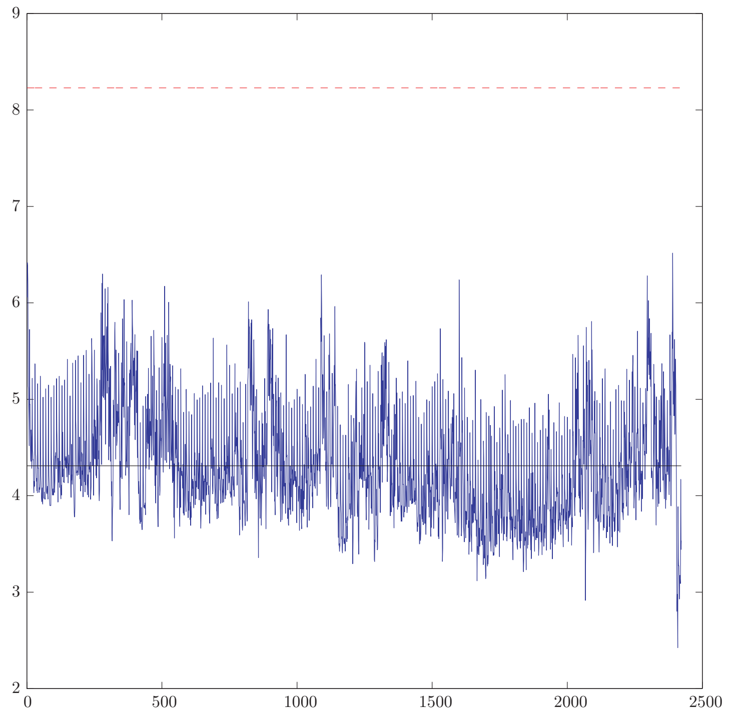

Figure 2.4: Complex environment: CE-MCL entropy measurement

T-shaped hall

Fig. 2.5 shows a typical CE-MCL estimation result for such environment. The presence of structural ambiguities along with the noisiness of laser readings, due to the nature of glass, make the localization problem more difficult. Despite the fact that the accuracy of the estimation is lower than the previous experiment, and the percentage of failures is markedly higher (mean value 40%, variance ±5%) the algorithm has been able to localize the robot, even under these critical conditions. Fig. 2.6 shows the measurement of entropy for this environment. As

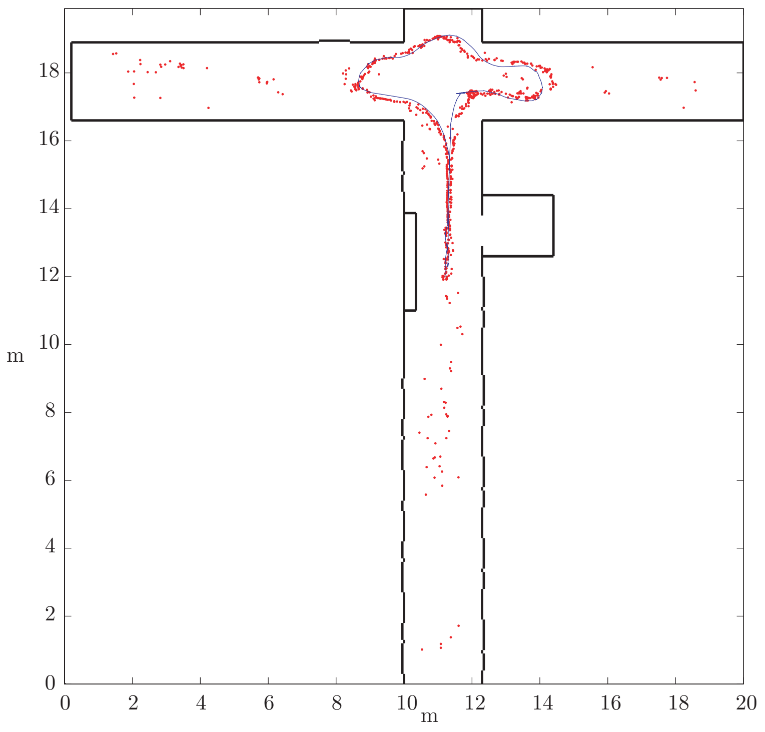

Figure 2.5: T-shaped environment: CE-MCL estimation result

in the previous experiment, the value settles around the median value, proving the algorithm's ability to maintain the diversity among the particles set.

Corridor

Fig. 2.7 shows a sequence of CE-MCL iterations for the corridor. This sequence of images describes the algorithm behavior between two resampling steps. In particular, when the first resampling occurs (a), the algorithm recognizes six clusters (the last one is the collection of noisy points) with a visible dispersion for the elements within the environment. The following iterations point out the CE-MCL tendency to centralize the hypotheses around the center of mass of each cluster. Moreover, after few time-steps, clusters are coarsely located along a line crossing the corridor. This deployment underlines the algorithm tendency to maintain only the most likely hypotheses after a full exploration of the environment. Two aspects of interest have been considered to study the convergence of particles within each cluster: a measure of similarity and the state variables variance. From a mathematical standpoint, both the state space variables analysis and the measurement of entropy give evidence of these considerations. Fig. 2.8 shows a

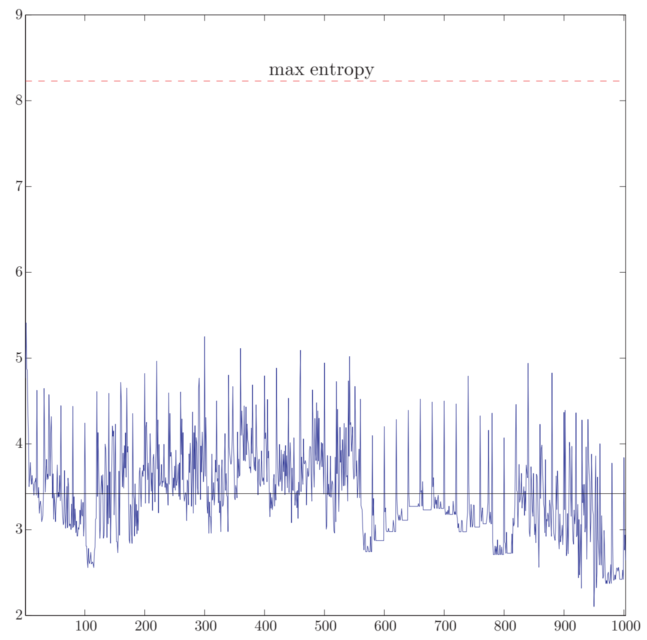

Figure 2.6: T-shaped environment: CE-MCL entropy measurement

typical variance trend within a cluster for all three state variables: peaks indicate the resampling effect, whereas slopes give evidence of the algorithm aptitude to make particles converge within each cluster. Fig. 2.9 exhibits a typical measurement of entropy within a cluster: the trend is similar to the previous one due to the relationship among these concepts, when restricted to a single cluster.

2.4 Considerations

In this chapter an enhanced Monte Carlo Filter has been presented to deal with the localization problem: the *Clustered Evolutionary Monte Carlo Filter* (CE-MCL). This algorithm has been conceived to overcome the classical Monte Carlo Filter drawbacks. This goal has been achieved taking advantage of an evolutionary approach and a clustering method. In particular, the former has been exploited to quickly find out local maxima, whereas the latter, being dynamical, helps to obtain an effective exploration of the environment. The ability to provide a smart partition set of the research space along with the guarantee to converge within each subset, make the algorithm able to solve the localization problem and maintain the multi-hypotheses. Note that the combined use of cluster+genetic offers several interesting advantages. At local level, being the size of research space smaller, the localization of the best solution is faster and the probability to stall on a suboptimal solution is lower. At global level, being the clustering dynamical and data-driven, an implicit parallelization of the research is possible and a better coverage of the environment is obtained, focusing the

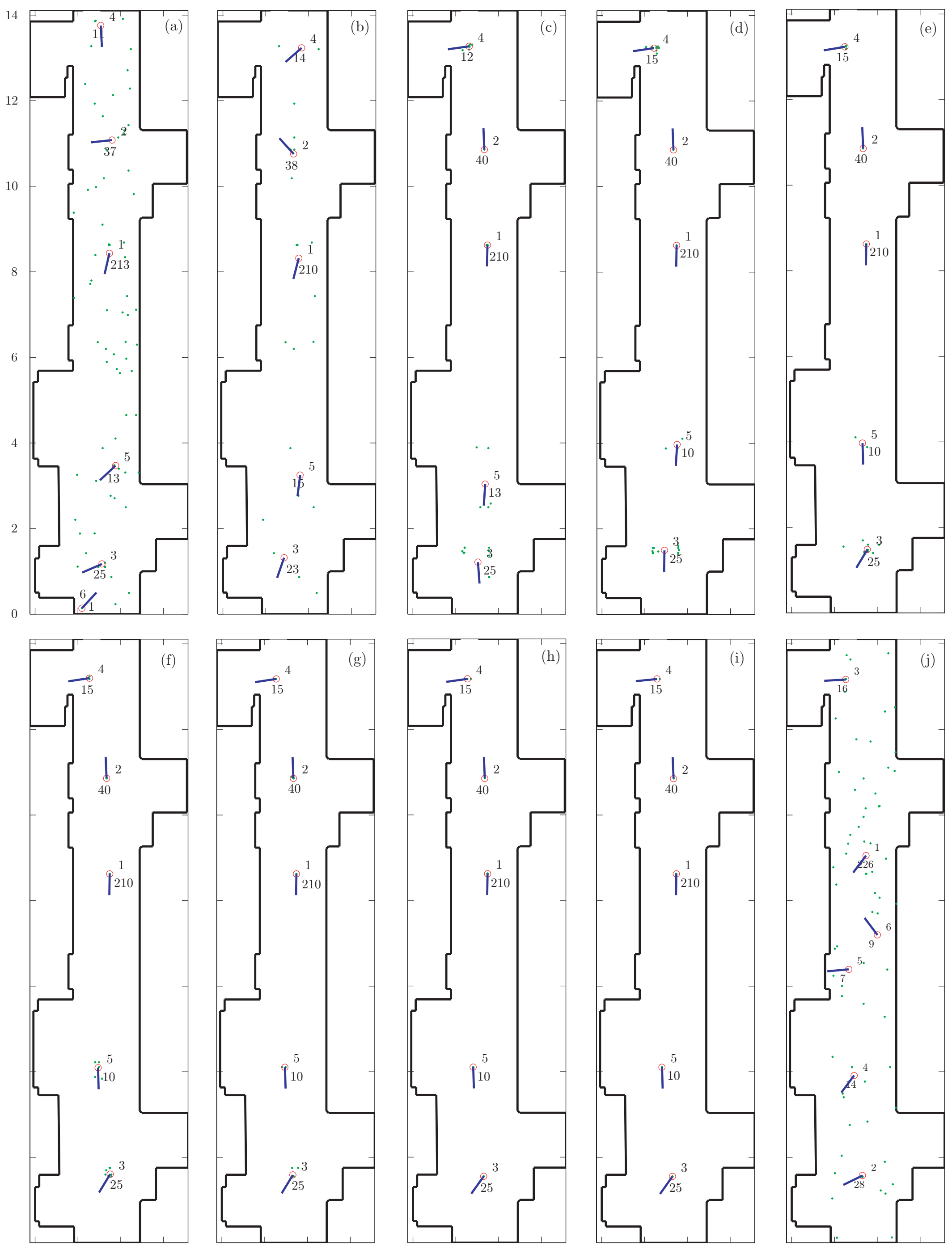

Figure 2.7: Corridor-like environment: CE-MCL sequence of iterations

attention where the probability to find out the real robot pose is higher. Exhaustive analyses have been performed on the robot ATRV-Jr manufactured by the IRobot, with the employment of several environments, to prove the effectiveness of the proposed algorithm. In particular, two different kinds of experiments have

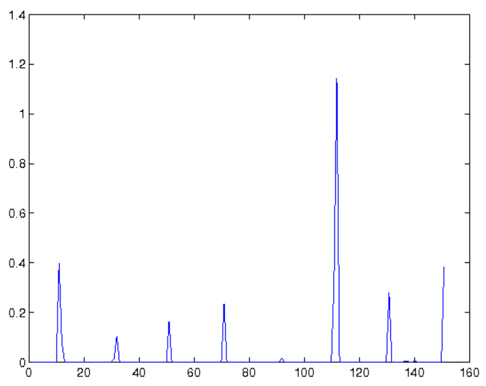

Figure 2.8: Corridor-like environment: CE-MCL variance

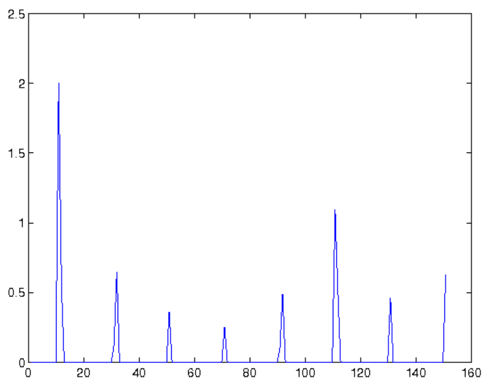

Figure 2.9: Corridor-like environment: CE-MCL entropy measurement

been considered: the first one has proved the algorithm ability to solve the global localization problem, even when a kidnap occurs; the second one has given evidence of the algorithm tendency to converge within each cluster and to guarantee an efficient exploration of the environment. Such analyses have shown the important role of the dynamical spatial clustering to provide an effective partition of the research space on which apply the evolutionary action. Therefore, the CE-MCL can find out local-maxima, guarantee a convergence to the most likely hypotheses, maintain the diversity among particles and localize the robot.

Chapter 3

A Spatially Structured Genetic Algorithm

In this chapter a novel approach based on spatially structured genetic algorithms is proposed. This approach takes advantage of the complex network theory for the spatial deployment of the population. In fact, modeling the search space with complex networks and exploiting their typical connectivity properties, results in a more effective exploration of such space. On the other hand, the introduction of spatial structures in evolutionary algorithms helps to create evolutionary niches. Since niches represent regions in which particular solutions are preserved, a natural way to maintain the multi-hypothesis is achieved. The approach originally developed for the single-robot context has been successively extended to deal with the multi-robot context. A technique for integrating the information exchanged by robots anytime they meet is proposed with the aim of extending their sensory capabilities.

3.1 Theoretical Background

3.1.1 Complex Networks

Complex Networks are graphs of nodes or vertices connected by links or edges, currently used to describe many natural or artificial systems: the brain, for instance, can be modeled as a network of neurons, and the Internet as a complex network of routers and computers linked by several physical means. From the beginning, complex networks have been investigated by the graph theory community, who proposed several models, such as regular and random graphs; since then several other communities have been interested in this topic. Today, a main research issue is to figure out the relationship between structural and dynamic properties of the networks.

Regular graphs, introduced to describe systems made of a limited number of nodes, were revealed to the research community to be inadequate with the appearance of large-scale networks. This has lead the community to focus their attention on random graphs. According to [120], once the probability p of having a connection among pairs of nodes is fixed, a random graph with N nodes and about $pN(N-1)/2$ links, can be obtained randomly selecting a pair of nodes and linking them with such probability p. This model has been extensively used since particular properties of complex networks, such as the small-world property or the scale-free one, have been discovered.

To better understand such properties, some basic concepts about complex networks have to be introduced: the *average path length*, the *cluster coefficient* and finally the *degree distribution*.

The *average path length* L of the network is the mean distance between two nodes, averaged over all pairs of nodes, where the distance between two nodes is defined as the number of the edge along the shortest path connecting them.

The *cluster coefficient* C of the network is the average of C_i over all nodes i, where the coefficient C_i of node i is the average fraction of pairs of neighbors of the node i that are also neighbors of each other.

The *degree distribution* of the network is the distribution function $P(k)$ describing the probability that a randomly selected node has exactly degree k, where the degree k is the number of links a node owns.

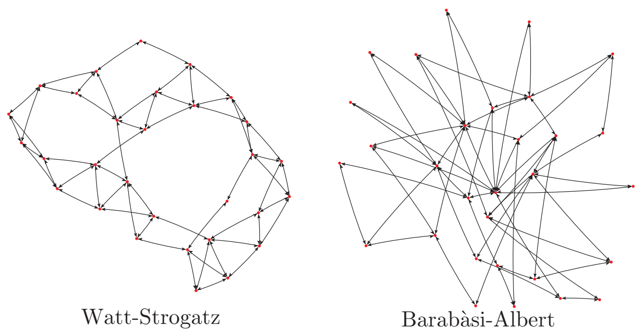

Figure 3.1: Watt-Strogatz and Barabási-Albert models with 30 nodes

From a formal point of view, regarding these basic properties, several complex network models can be correctly defined. Regular graphs, for instance, are characterized by a high cluster coefficient, approximately $C \cong 3/4$ and a large average path tending to infinity as $N \to \infty$. Random graphs have a low cluster coefficient, approximately equal to the probability p defined above, and a short average path $L_{aver} \cong lnN/(pN)$. In [154] the *Small-World* model is proposed to better describe real systems. It shows properties of both the regular and random graphs, such as a high cluster coefficient and a short average path, underlining the fact that, in reality, the circle of acquaintances of people is not only restricted to their neighbors. In [15] the *Scale-Free* model is presented. This model, relying on the power-law degree distribution, overcomes the limitations of the previous ones through a hierarchical description of nodes. As a consequence, in a scale-free network preferential attachments are possible. This model, for instance, turns out to be very useful to describe airline routing maps.

Two examples of the above described networks are reported in Fig. 3.1; for a complete overview of complex networks the [153] is suggested.

3.1.2 Spatially Structured Genetic Algorithms

Spatially Structured Genetic Algorithms (SSGAs) represent a special class of Genetic Algorithms (GAs), which are described in Appendix B, where the population is spatially distributed with respect to some discrete metric. SSGAs are characterized by different properties than standard GAs, such as the capability to preserve the diversity or to create evolutionary niches.

SSGAs can be properly described by exploiting the graph theory. In this context, given a population $P = \{p_1, \dots, p_n\}$ and a graph $G = (V, M)$, where $V = \{v_1, \dots, v_n\}$ is the set of vertices and $M = \{(i,j) = 1 : \exists \; \textit{link between } i \textit{ and } j\}$ is the incidence matrix, a Spatially Structured Genetic Algorithm can be obtained by defining an isomorphism $\mathcal{F}(\cdot)$ from P to V so that:

$$\mathcal{F} : P \rightarrow V \tag{3.1}$$

Therefore, a SSGA is a variation of a Simple Genetic Algorithm (described in Appendix B) where selection is performed by exploiting the topology of the graph underlying the population in spite of using the roulette wheel.

Note that, this formalization was already used to introduce the *graph based genetic algorithms* in [9]. Here, the idea was to prevent the loss of diversity by imposing geographical structure to the population to limit choice of crossover partners. Finally, a more comprehensive treatment of this class of genetic algorithms is given in [146]

3.2 The proposed spatially structured genetic algorithm over complex networks

The proposed SSGA provides a framework for both single robot and multi-robot localization problem. It takes advantage of the complex network theory for the deployment of the population. Giving such a structure to the population leads to several interesting advantages, such as the capability to carry on the multi-hypothesis paradigm.

In this context, a chromosome encodes the state of the robot, represented through its position and orientation (x, y, θ). From an algorithmic standpoint, a SSGA reflects the classical SGA schema described in Appendix B with a specialization regarding the structure of the population. In the initialization step, a population is randomly generated and its individuals are connected to obtain a complex network, typically small-world and scale-free. At each epoch k, the population evolves dynamically, according with the kinematic model of the robot and the input applied, maintaining the topology of the network. Such a topology is exploited during the selection phase, which is trivially realized picking up all the pairs of linked elements. In the reproduction stage the mating rules represented in Table 3.1 are used to choose the probabilistic transition operator.

To apply the mating rules, each individual is labeled with the tag *high* or *low*, computed by means of a comparison between the fitness of the individual itself and the average fitness of the population.

Table 3.1: Mating rules

Node 1	**Node 2**	**Action**	**Basic principles**
Low	Low	Both self-mutate	Mutation
High	Low	Node 2 is replaced with a Mutation of Node 1	Elitism and Mutation
Low	High	Node 1 is replaced with a Mutation of Node 2	Elitism and Mutation
High	High	The lower is replaced with the Crossover on the two	Elitism and Crossover

Regarding the probabilistic transition operators: *crossover* picks up two elements and performs a convex combination of them with probability p_{cross}, while *mutation* picks up an element and modifies its chromosoma, inversely proportional to its fitness, with probability p_{mutat}.

3.2.1 Independent Evolution

Anytime a robot is moving and no other one is within its communication range, the only information available is the data coming from its sensors. Therefore, each robot performs an independent evolution computing a measure of similarity between data coming from a real robot and the expected one computed for a given hypothesis. Specifically, the fitness function of an individual i weights the compliance between the exteroceptive measurements (z_k) retrieved by the robot and the ones ($\hat{z}_k$) expected by the individuals itself

$$\Phi_i(z_k, \hat{z}_k) = \frac{1}{L} \sum_{j=1}^{L} \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z_k^j - \hat{z}_k^j)^2}{2\sigma}} \tag{3.2}$$

where σ is the confidence of the sensor and L is the number of measurements considered. The pseudo-code in algorithm 3 shows a possible implementation schema for a generic step k of independent evolution for the proposed SSGA.

3.2.2 Cooperative Evolution

The same approach holds when considering multi-robot localization. In this case, although for each robot a population is initialized and lets evolve independently, a collaboration can be set-up any time when robots meet. The key idea is to integrate the observations coming from the components in such a way that the sensory system capability of each robot is extended. In order to achieve that, the relative position and orientation of the robots in the team are assumed available along with the sensor data, while the fitness function is augmented in the following

Algorithm 3: The proposed SSGA - iteration k

```
Data: Population of size $n$: $\{V = \{p_{j,k}\}, M\}$, $\Phi(\cdot)$, $e(\cdot)$
Result: $V = \{p_{j,k+1}\}$

/* Average Fitness Evaluation                                   */
1  $\Phi_{aver} = \sum_{i=1}^{n} \Phi(p_{i,k})/n$
/* Incidence Matrix Selection                                   */
2  for $i=1$ to $n$ do
3      for $j=i$ to $n$ do
4          if $M(i,j) = 1$ then
5              switch $Compare(\{\Phi(p_{i,k}), \Phi(p_{j,k})\}, \Phi_{aver})$ do
6                  case High-High
7                      if $\Phi(p_{i,k}) > \Phi(p_{j,k})$ then
8                          $p_{j,k} = Crossover(p_{i,k}, p_{j,k})$
9                      else
10                         $p_{i,k} = Crossover(p_{i,k}, p_{j,k})$
11                     end
12                 case High-Low
13                     $p_{j,k} = Mutation(p_{i,k})$
14                 case Low-High
15                     $p_{i,k} = Mutation(p_{j,k})$
16                 case Low-Low
17                     $p_{i,k} = Mutation(p_{i,k})$ $p_{j,k} = Mutation(p_{j,k})$
18             end
19         end
20     end
21 end
22 $\{p_{j,k+1}\} = \{p_{j,k}\}$
/* Kidnap Detection                                             */
23 if Kidnap Condition then
24     Spreading Action
25 end
```

way

$$\Phi_i(z_k, \hat{z}_k) = \frac{1}{L}\sum_{j=1}^{L} \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z_k^j - \hat{z}_k^j)^2}{2\sigma}} + \sum_{r=1}^{R} \frac{1}{L_r} \sum_{l=1}^{L_r} \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z_k^l - \hat{z}_k^l)^2}{2\sigma}} \tag{3.3}$$

where R is the number of the robot of the team in the viewing area and L_r the number of sensor of the r–th robot. The second addendum weights the compliance of the measurement of the team formation with respect to the formation replicated around the i-th individual. In this way the localization algorithm results completely distributed and collaboration is possible even when robots move. Cooperation turns out to be fundamental in reducing the perceptual aliasing, as the more data available the higher the probability to converge to a single location. It is well known that a genetic approach helps to maintain a population of multiple hypotheses. In particular, SSGA, due to their convergence proprieties, usually maintain equally probable hypotheses and, in presence of sufficient and not ambiguous information, converge to a neighborhood of the solution. This fast takeover is related to the structure of the space of interactions and can be exploited monitoring the formation of a single cluster.

Although the algorithm is able to solve the global localization problem with a proper random initialization, an additional strategy, able to perceive when a kidnap occurs, is required in order to spread the population over the search space again and then re-localize the robot. In this work, the fitness function $\Phi(\cdot)$ and the edge function $e(\cdot)$, i.e., the fraction of potential mating couples (couples with different genotypes) over the network (the number of links) [9], are used to trigger the spreading action. The edge function gives an evaluation of the dispersion of the population. Specifically for a well-localized robot, a high percentage variation of the fitness along with a considerable dispersion of the population (a high value of the edge function), are reliable symptoms of a kidnap.

3.2.3 Computational Complexity

The analysis of the computational complexity is important to evaluate the capability of an algorithm to run online. For this reason, a detailed theoretical investigation along with an empirical validation of the obtained results has been performed. The pseudo-code proposed in Algorithm 3 presents two nested loops in which the dominant operation is the fitness evaluation whose complexity for each element of the population is $\mathcal{O}(M)$, where M is the number of segments describing the environment. As a consequence, the overall complexity for this naive implementation results in $\mathcal{O}(M \cdot N^2)$, where N is the size of the population. However, a more effective implementation can be provided by observing that:

- The fitness value for each element of the population can be stored during the evaluation of the average fitness function (algorithm 3 - line 1).
- A mating rule is applied only when a link is available between two elements (algorithm 3 - line 4).

This way the complexity can be significantly lowered. In fact, pre-evaluating the fitness and storing it reduces the complexity of the dominant operation within the nestled loop to a constant factor $\mathcal{O}(1)$. On the other hand, the two loops can be replaced with a vector indexing all the couples with a link. Since for the considered topologies the number of links is proportional to the size of the population, i.e., $(k-1) \cdot N/2$ with k node degree of each node, the over complexity of the algorithm is reduced to $\mathcal{O}(k \cdot N \cdot M) = \frac{(k-1) \cdot N \cdot M}{2}$.

The first element is the cost of scanning all the couples with a link, while the second is the cost of evaluating the fitness for the whole population.

3.3 Performance Evaluation

The proposed Spatially Structured Genetic Algorithm has been extensively investigated in both simulated environment and with real robot data. Real experiments allow to validate the algorithm in different real-world contexts, while simulations are useful for tuning the algorithm parameter and for validating the multi-robot extension. In the following, some results of the experimental campaign are presented. The exposition is organized in two paragraphs, the first analyzes the behavior of the algorithm for single robot localization while the second reports some results for multi-robot.

3.3.1 Problem Settings

Computer Simulations. Simulations have been carried out in a framework developed under Matlab by the authors. This framework provides different kinematic models for the robot, such as the unicycle model, as well as an emulation for several sensors such as a laser rangefinder. Moreover, the environment is described by a set $\mathcal{M}$ of segments. This framework supports both a complete simulated context as well a test-bed to run data coming from a real robot. These two different operative modalities turn out to be very useful, both to test the correctness and the effectiveness of the algorithm. Details about robot, sensors and environment modeling are provided in Appendix A.

Real Robot Context. Experiments have been carried out on the mobile platform ATRV-Jr manufactured by iRobot. It is a skid steering vehicle mainly designed to operate in outdoor environments. The ATRV-Jr has 4 wheels differentially driven by 2 DC motors: the motion is achieved by a differential thrust on the wheel pairs at the opposite sides. The mobile robot is equipped with 17 sonar rangefinders, a laser scanner (Sick LMS-220), an inertial platform (Crossbow DMU-6X), and a GPS receiver (Garmin GPS35-HVS). The sensory system is connected to the ATRV-Jr's on board PC (Pentium II, 350 MHz) running Linux, through serial ports on a Rockeport multiserial port card. The robot is delivered with a software development environment called MOBILITY, which provides full access to the software servers available on the mobile platform. Each server is assigned to control a specific hardware component (sensors and actuators). In this way all interfaces are reachable from the network exploiting a CORBA interface.

3.3.2 Single robot localization

Simulations

The first simulation characterizes the topological model adopted for the network with respect to the persistence of several evolutionary niches, corresponding to several hypotheses. Such analysis has pointed out, for the Watt-Strogatz model, that a network degree equal to 3 and a rewiring probability of 0.1 are a satisfactory set of parameters; a similar analysis, reported in [70], shows that for these values, such a network shows a high clustering coefficient and a small average path length. This condition increases the selection pressure (the speed of convergence is high but not maximum) along with the persistence of niches. For the scale-free

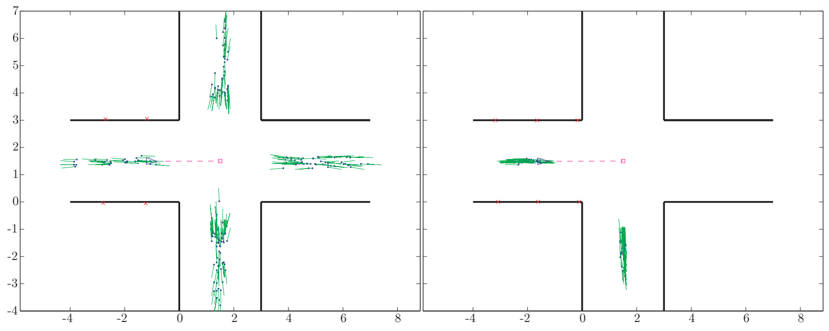
Figure 3.2: Small-world (left) and scale-free (right) networks

Barabási-Albert model the behavior is slightly different due to the presence of *hubs*, i.e., highly connected nodes. Scale-free networks have a shorter takeover time (the time it takes for the single best individual to conquer the whole population), that could lead the SSGA to converge too rapidly towards an incorrect solution in a noisy environment, as it can be see in Fig. 3.2. For this reason the experiments presented below have been carried out using Watt-Strogatz model.

Robot in Real Environment

The ATRV-Jr was put into two different indoor office-like environments:

- Corridor.
- Lobby.

The Corridor The robot, starting in S, moves downward and, after a U-turn around iteration 100, is kidnapped in K (iteration 120) to appear in N where it goes on until E (Fig. 3.3). In the first 15 iterations, the network, randomly spread, forms some clusters, one of these will survive and after 20 iterations a single cluster remains as winner. Consequently, the tracking of the correct position is accomplished with very low errors: the best hypothesis, i.e., the individual of the population having the best fitness, has a mean Euclidean distance of about 15 cm from the real robot location. During the U-turn (iterations 95–110) some measures, which do not correctly fit with the environmental model, along with the inaccuracy of the odometric prediction, will result in an inaccurate tracking. The fitness function is also affected by this situation, but the small increasing of the edge function shows how the network begins to explore new solutions in the surroundings to recover the error.

The kidnap sensing strategy, as previously shown, relies on the use of the fitness function along with the edge function. In particular, an analysis has been carried out to find out the relationship between the kidnap event and the variation of these functions. According to the experimental results, when a kidnap occurs,

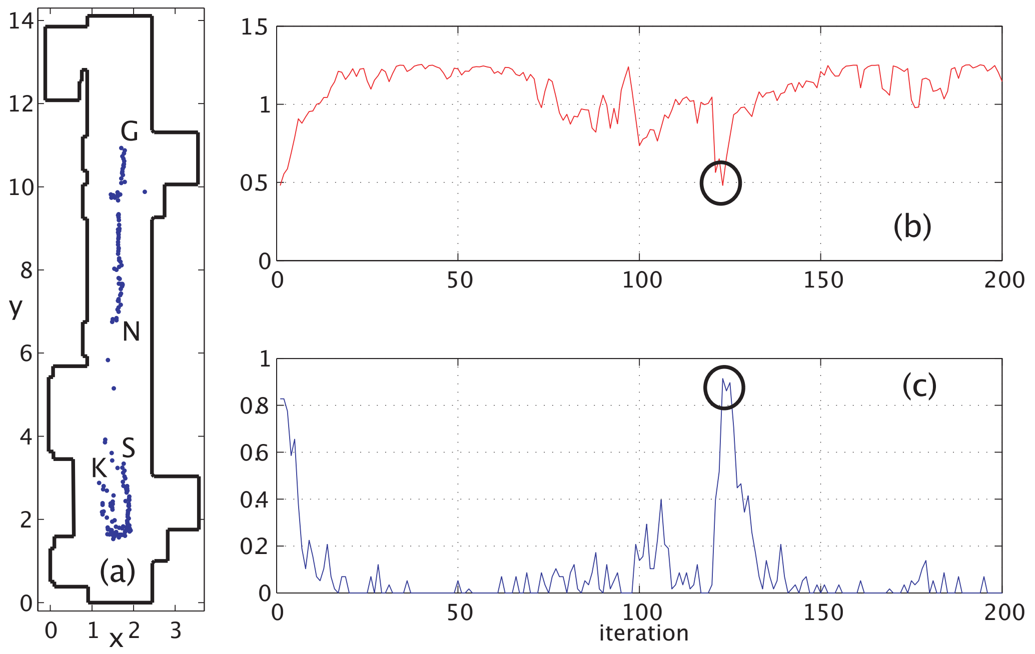

Figure 3.3: (a) Map of the first environment and path of the robot (S:start, K:kidnap point; N:new start; E: end of path) - (b) Fitness function - (c) Edge function.

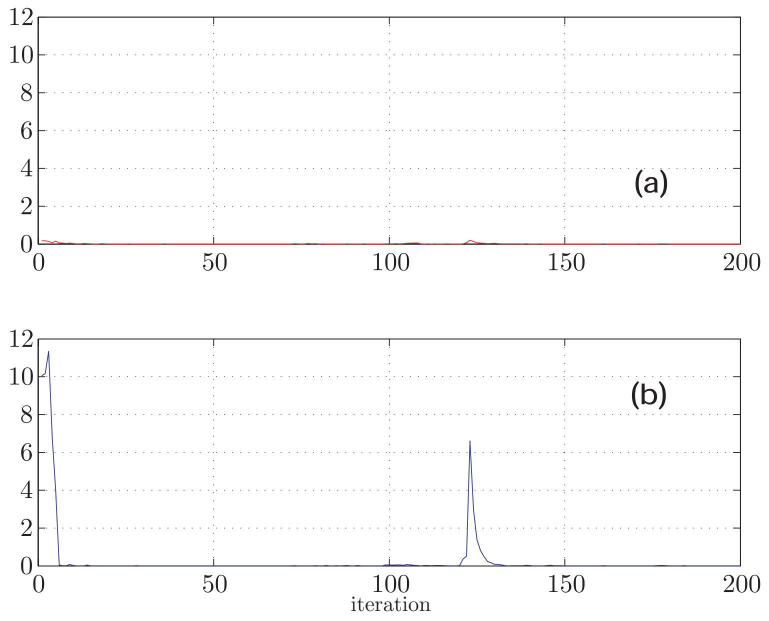

Figure 3.4: First Environment. (a) Dispersion along x axis - (b) Dispersion along y axis.

the fitness function drastically decreases, whereas the edge function considerably increases because of the probabilistic operators effect. Consequently, a variation of the fitness function along with an increase of the edge function has been used as a threshold. In Fig. 3.3 (b) and (c) this situation is clearly represented. Note that after the kidnap, the dispersion along y axis increases as long as the error along y. However, as shown in Fig. 3.4, the error is quickly recovered in few iterations as the network starts to correctly track the robot.

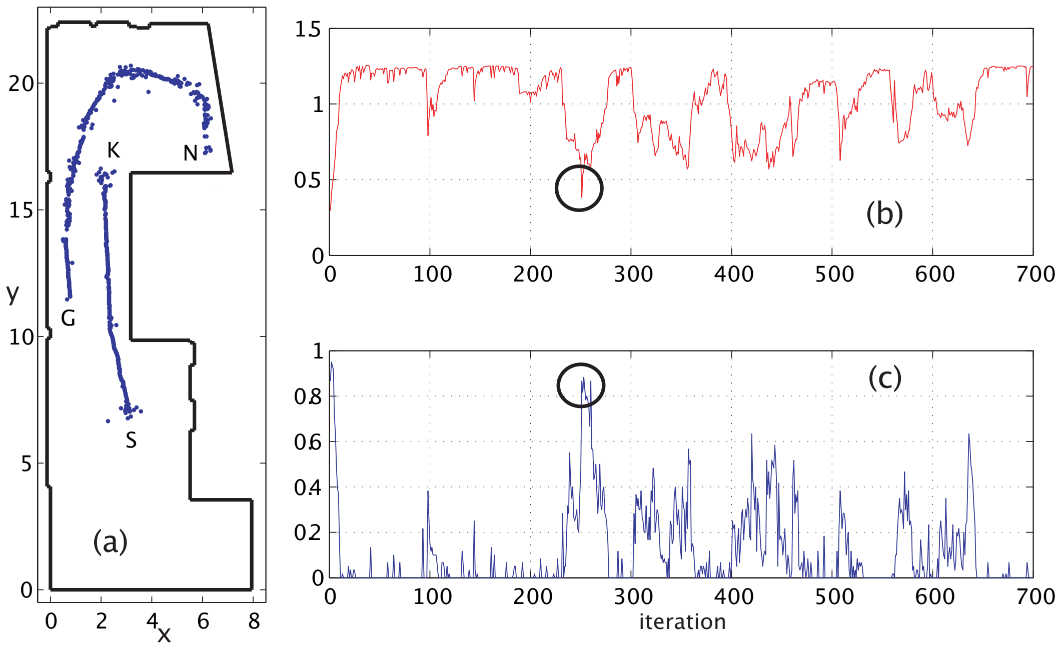

Figure 3.5: (a) Map of the second environment and path of the robot (S:start, K:kidnap point; N:new start; E: end of path) - (b) Fitness function.

The Lobby The robot, starting in S, moves upward and, after a straight path, is kidnapped in K (iteration 250) to appear in N where it goes on until E (Fig. 3.5) . As in the previous experiment, after an initial random deployment of the network, few clusters are maintained until the one representing the real position remains as a winner. Note that, in average both values of the fitness function and edge function are respectively lower and higher compared to the previous experiment. This can be explained by the fact that the map available for this area was less accurate. However, it turned out to be a good scenario to test the robustness of the kidnap sensing strategy. In fact, although the two indexes were more “noisy”, the kidnap event has been properly recognized.

An increment of the dispersion along both axes is experienced due to the kidnap event, . In addition, Fig. 3.6 shows the trend of the dispersion along both axes. I

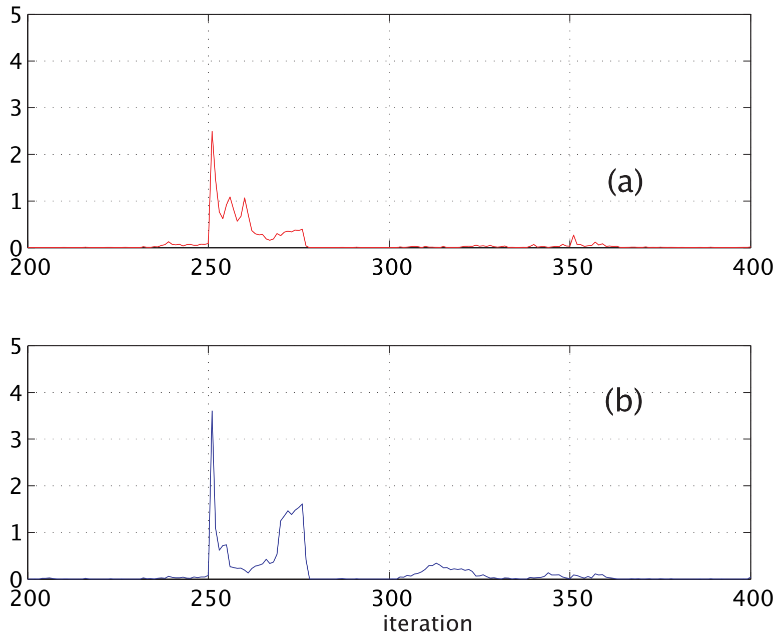

Figure 3.6: (Second Environment. (a) Dispersion along x axis - (b) Dispersion along y axis.

3.3.3 Multi-robot localization

The first simulation has been carried out in a simple environment with only one wall and two landmarks. It has been intentionally exploited as the information coming only from laser rangefinders are not sufficient to fully distinguish among all hypotheses.

Fig. 3.7 shows the behavior of the algorithm at some time-steps, when the collaboration among robots is *deactivated*. Robots are represented by triangles and populations are reported as dots with orientation given by small segments. In (a), after only 4 iterations, each robot recognizes only one landmark, as a consequence each population tends to dispose along a circle whose radius is the measured distance. Note the ability of the algorithm to expand all hypotheses to cover all possible locations. In (b), with regard to robot 1, hypotheses tends to dispose along a line as soon as the wall is recognized. In particular, the shape of this line is given by the integration of data related to the wall with the observations of the landmark. In (c), after 48 iterations, the situation after few other iterations is shown: as predictable, robots cannot fully localize themselves due to the lack of information available.

Fig. 3.8 shows the behavior of the algorithm at the same time-steps, when the collaboration among robots is *activated*. Robots are assumed to be able to communicate to each others within a range of 5 m. In this case, even after few steps (4 iterations) the situation is less ambiguous (a). In fact, in (b) (iteration 24) robots are already localized, even though with a perceptible uncertainty,

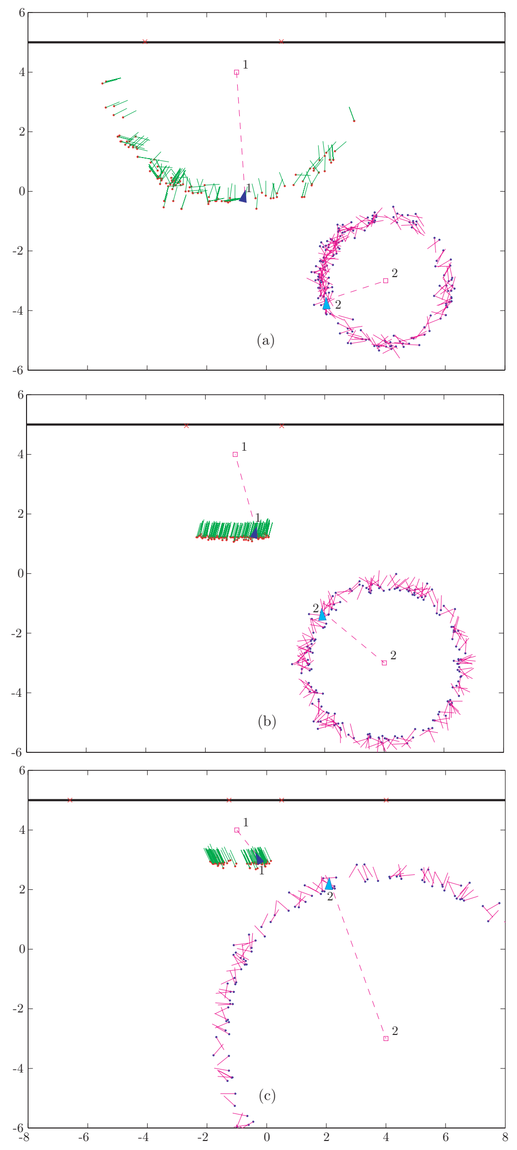

Figure 3.7: First simulation in independent mode: iterations 4 (a), 24 (b) and 48 (c)

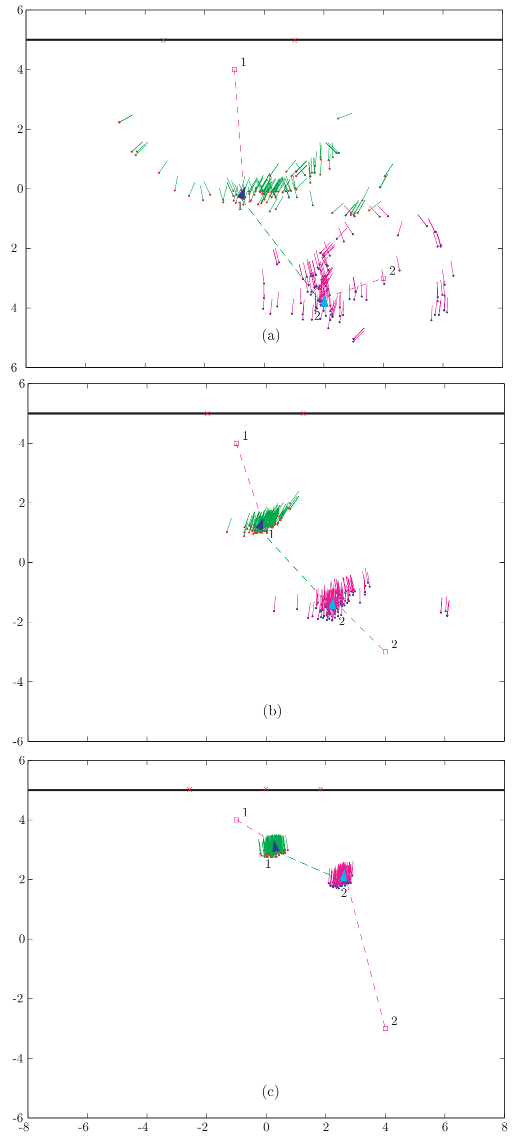

Figure 3.8: First simulation in collaborative mode: iterations 4 (a), 24 (b) and 48 (c)

underlined by the hypotheses arrangement. In (c), having the cooperation been activated for long enough, robot are fully localized proving that the cooperation can better exploit data.

The second simulation has been carried out in a highly symmetrical environment. Also in this case, information coming only from laser rangefinders are not adequate to fully distinguish hypotheses.

Fig. 3.9 shows the behavior of the algorithm at some time-steps when the collaboration among robots is *deactivated*. At the beginning, as no prior information is available, hypotheses are uniformly spread over the whole environment. Afterwards, according to data coming from sensors, some regions turn out to be

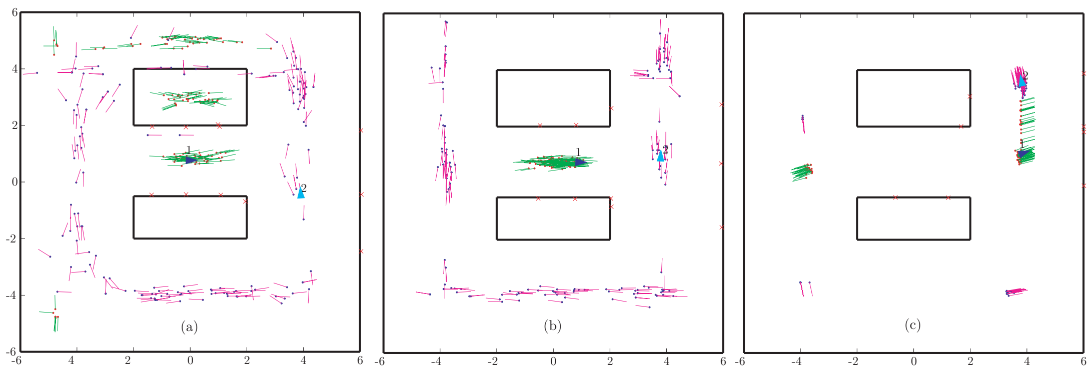

Figure 3.9: Second simulation in independent mode: iterations 12 (a), 24 (b) and 48 (c)

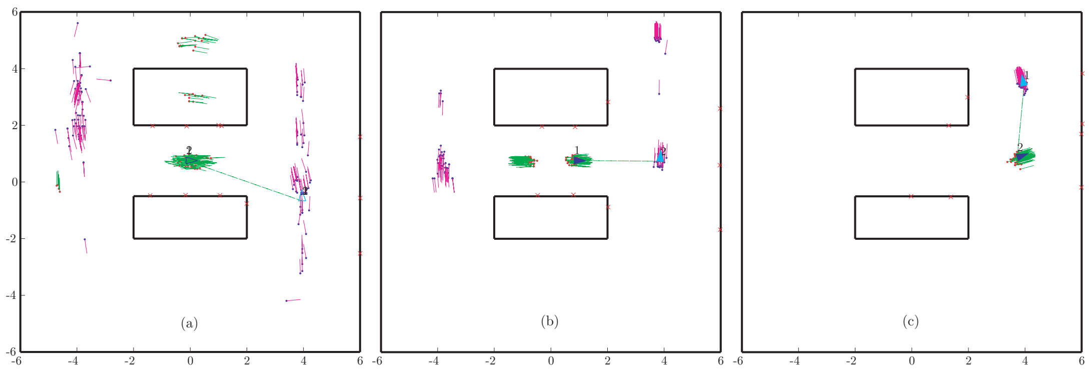

Figure 3.10: Second simulation in collaborative mode: iterations 12 (a), 24 (b) and 48 (c)

more likely than others. Therefore, individuals tend to cover these areas. However, as expected, robots cannot fully localize themselves relying only on the laser rangefinder data because of the nature of the environment.

Fig. 3.10 shows the behavior of the algorithm at the same time-steps when the collaboration among robots is activated. Robots are assumed to be able to communicate with each other within a range of 5 m. Also in this case, as no prior information is available at the beginning, hypotheses are spread uniformly over the whole environment. However, due the cooperation between the two robots, a different behavior can be noticed. In fact, extending the perceptual capability of a robot, a better "sight" of the environment is obtained. Therefore, some structural ambiguities can be easily overcome.

The third simulation has been carried out considering three robots. Fig. 3.11 shows the 55-th iteration: in (a) the algorithm behavior when the cooperation is deactivated is reported, while in (b) the same situation with the support of the cooperation is shown. Also in this context, it can be easily noticed as extending the perceptual capability of a robot improves its localization. In particular, due to the nature of the environment, robots 2 and 3 take advantage of the collaboration to disambiguate their position, while robot 1 can localize itself simply exploiting the laser rangefinder data. Moreover, the framework allows different perceptual

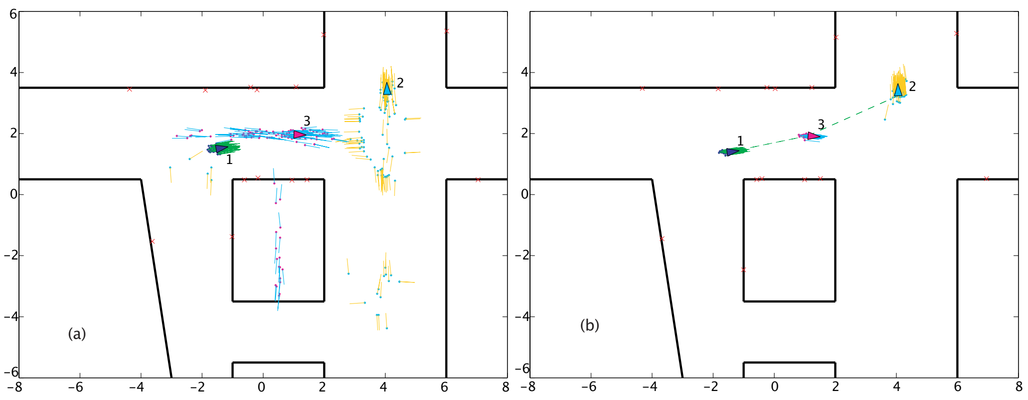

Figure 3.11: Third simulation in independent (a) and collaborative (b) mode: iteration 55

extensions to exist. In fact, while robot 3 extends its perception using data coming from both robots 1 and 2, these robots exploit only data coming from one robot.

3.4 Considerations

In this chapter a novel approach based on spatially structured genetic algorithms for mobile robot localization has been proposed. It takes advantage of the complex network theory for the deployment of the population in order to more quickly find out the optimal solution. In fact, modeling the search space by means of complex networks results in a more effective exploration. In addition, giving such a structure to the population leads to several interesting advantages such as the capability to create evolutionary niches. Since niches are regions in which particular solutions are preserved, a natural way to carry on the multi-hypothesis is obtained. This approach has been successively extended to deal with the multi-robot context. The idea is to exploit the underlying collaborative nature of multi-robot systems so that the perceptual aliasing can be reduced, as the more data available the higher the probability to converge to the exact location. This is achieved by integrating observations coming from several robots so that the sensory system capability of each robot is extended. In order to realize that, relative position and orientation of the robots are exchanged along with the sensor data. Real experiments have been carried out to prove the algorithm capability to deal with the global localization problem and validate the strategy that has been proposed to *sense* when a kidnap occurs, while extensive simulations have been performed to validate the strategy proposed for the multi-robot context. According to the simulation results, the collaborative technique significantly reduces the perceptual aliasing, particularly when the interaction with the environment does not provide enough information.

Chapter 4

A Bacterial Colony Growth Framework

In this chapter, a new-biology inspired approach is proposed. The idea is to exploit models of species reproduction to provide a suitable framework for maintaining the multi-hypothesis. The Bacterial Colony Growth Algorithm(BCGA) provides two different levels of modeling: a background level that carries on the multi-hypothesis and a foreground level that identifies the best hypotheses according to an exchangeable strategy. In addition, a collaborative policy for the multi-robot context is proposed. Collaboration among robots is obtained by exchanging sensory data and their relative distance and orientation. This information is integrated into the framework in such a way that the convergence aptitude is enhanced redefining the condition (nutrient or noxious) of the environment.

4.1 Models of Species Evolution

The evolution of species has been modeled mathematically with different approaches. Historically, systems of deterministic differential equations suitable for large population dynamics were the first proposed [104] [148] [150]. More recently, in order to model in-vivo reactions such as metabolic or gene regulations, stochastic differential equations [38] and lattice gas automata using Monte Carlo Algorithms [137] have been introduced.

4.1.1 Competitions and Cooperation among Cohabitant Species

The basic model (apart from the Malthusian one [104]) for describing species evolution is the *logistic* model, introduced by Verhulst [148]. In this model, the Malthusian natality factor r is a linear function of the population numerousness $N(t)$

$$r(N(t)) = r_0 - r'N(t) \tag{4.1}$$

where r_0 and r' are opportune positive constants typical of the population. From the above equation, it follows that the population grows when the natality rate is positive and the numerousness is not too large (below r_0/r'). Eq. (4.1) describes an *auto-regulation* linked to the numerousness in the natality process. Introducing eq. (4.1) in the Malthusian equation $\frac{dN}{dt} = RN(t)$, where $R = r - m$ is the growth rate, m being the mortality factor, the logistic equation is defined as:

$$\begin{aligned} \frac{dN(t)}{dt} &= (r_0 - r'N(t) - m)N(t) \qquad (4.2) \\ &= R_0 N(t)\left(1 - \frac{N(t)}{K}\right) \end{aligned}$$

where $R_0 = r_0 - m$ is the population growth rate in the absence of intra-specific competition and the ratio $K = \frac{R_0}{r'}$ is the *carrying capacity.* The solution of this differential equation is:

$$N(t) = \frac{K}{1 + (\frac{K}{N(0)} - 1)e^{-R_0 t}} \qquad (4.3)$$

In an evolutionary framework, different species compete for the same resources in order to survive. In other words, the growth of different species is limited by a common factor. Supremacy (survival) of one species over the others is determined by natural selection. The logistic equation previously introduced for a single species can be properly modified to model such competition. For two species, assuming the overall numerousness $((N_1(t) + N_2(t)))$ as the common factor, the following equations can be derived:

$$\begin{aligned} \frac{dN_1(t)}{dt} &= \left(1 - \frac{(N_1(t) + N_2(t))}{K_1}\right) R_1 N_1(t) \qquad (4.4) \\ \frac{dN_2(t)}{dt} &= \left(1 - \frac{(N_1(t) + N_2(t))}{K_2}\right) R_2 N_2(t) \qquad (4.5) \end{aligned}$$

Subsequently, the *predator-prey* was introduced by Volterra and Lotka [150]. Here the authors consider an environment composed by two populations in which predators eat prey.

$$\begin{aligned} \frac{dH(t)}{dt} &= (a - bP(t))H(t) \qquad (4.6) \\ \frac{dP(t)}{dt} &= (kH(t) - c)P(t) \qquad (4.7) \end{aligned}$$

Depending on the constant values, the populations can present different behaviors, including periodic ones.

Competition and *cooperation* can be modeled in a more general framework, where different species are living in the same environment. Consider again a biological system comprised of two populations P_1, P_2 and a limited resource that both populations need. To use the resource, P_1 and P_2 begin to compete. Let's assume that if one population extinguishes, the other one grows according to logistic law $\frac{dP_i(t)}{dt} = a_i P_i(t) - b_i P_i^2(t)$. Moreover, in the cohabitation, there is an encounter term $cP_1(t)P_2(t)$ that has a control effect. The evolution is then described by:

$$\frac{dP_1(t)}{dt} = (a_1 - b_1 P_1(t) - c_1 P_2(t))P_1(t) \tag{4.8}$$

$$\frac{dP_2(t)}{dt} = (a_2 - b_2 P_2(t) - c_2 P_1(t))P_2(t) \tag{4.9}$$

where a_is are the growth rates, b_is are the *intra*-specific competition coefficients, c_is are *inter*-specific competition coefficients.

In a similar way, a model for the cooperation can be designed in which a population will extinguish if the other one is lacking. In this model, the genetic evolution of species (either by sexual reproduction, i.e. genetic mixtion, or mutation) is not explicitly defined. In order to handle this, several solutions can be presented. One is to allow new speciation and consider different evolved genetic strains of the same species as different competitive-cooperative populations. Another is to introduce correction terms in the reproduction rates as a result of the overall evolution of a species (interpreted as modified replication capacity).

4.2 The Bacterial Colony Growth Algorithm

A major issue of the global localization problem is maintaining a set of hypotheses about the robot pose until a reasonable confidence level of estimation is reached. The *Bacterial Colony Growth Algorithm* takes this issue into account. As a result, it provides two levels of modeling:

- The *Background Level* that provides a suitable framework for modeling and carrying on the multi-hypothesis.
- The *Foreground Level* exploits several exchangeable strategies to track the robot pose.

4.2.1 Background Level: Multi-Hypothesis Modeling

The models of species reproduction introduced above can be effective in describing and maintaining the multi-hypothesis. In this context, a population of hypothetical robots is considered. Each robot is seen as a *bacterium* in a biological environment, say *Escherichia Coli*, which reproduces asexually. One interesting phenomenon observed in the unicellular organisms is the *chemo-taxis response*, in which the cellular movement is oriented towards or away from a chemical compound. Mobile bacteria as E. Coli swim towards areas with a higher concentration of nutrient compounds like sugars (attractors) or amino acids, and away from higher concentrations of noxious compounds (repulsors), so that its motivation is similar to that of a particle in a vector field based on a gradient method. Clearly, the environment is composed of different areas characterized by compounds and concentrations which vary over time. Another interesting characteristic of protozoa is that they can form colonies and aggregate in specific regions. Conversely, when no favorable conditions are present, they wait latently for better times to reproduce (*bet-hedging*).

In the mobile robot localization context, the nutrient areas represent regions where the measurements $\overrightarrow{m}_{r_t}$, provided by the real robot, match with some of the population estimated measurements $\overrightarrow{m}_{p_t}$s (and bad matches define noxious areas). Moreover, the kinematic model allows bacterial movement. At the same time, attractive and repulsive areas change dynamically according to the real robot movements. In the nutrient areas, the bacteria (robots) can reproduce and form colonies (clusters of robot hypotheses), whose growth is limited by the total resources of the environment and by the colony size. Thus, a natural way of maintaining the multi-hypothesis is achieved. In addition, the growth limitation curbs the unbounded growth of the best hypotheses as well as the extinction of other small-medium size colonies.

Specifically, when a bacterium is in a nutrient area, its chances of reproducing and forming a colony are higher while its replication chances are lowered by over-population. Moreover, if the nutrient area is shifting somewhere else, the colony first tries to expand slightly (dispersion), then starts to disintegrate if nutriment is no longer available (the environment becomes noxious), as can be clearly observed when a kidnap occurs. Finally, when attractive areas are unavailable or unreachable, the bacteria become latent and stop reproducing, wandering until suitable conditions are found. The bacterial colony growth algorithm (BCGA)

Algorithm 4: Bacterial Colony Growth Algorithm

```
Data: $P_t = \{p_{1,t} \dots p_{N,t}\}$
Result: $P_{t+1} = \{p_{1,t+1} \dots p_{N,t+1}\}$
1  $i = 1$;
2  while $(i \leq N)$ do
3      latency flag $l = TRUE$;
4      $j = 1$;
5      while $(j \leq N \wedge i \leq N)$ do
6          generate $r \in \mathcal{U}[0,1]$;
7          calculate $f_1(\overrightarrow{m}_{p_{j,t}}, \overrightarrow{m}_{robot_t}) = n \in [0,1]$;
8          calculate $f_2(p_{j,t}, P_{t+1}) = d \in [0,1]$;
9          if $(r < n \cdot (1-d))$ then
10             $p_{i,t+1} = reproduction(p_{j,t}, n)$;
11             $i = i + 1$;
12             $l = FALSE$;
13         end
14         $j = j + 1$;
15     end
16     if $(l = TRUE)$ then
17         $p_{i,t+1} = betHedging(P_t)$;
18         $i = i + 1$;
19     end
20 end
```

is shown in detail in Algorithm 4. The reproduction policy for each bacterium-robot is driven by both the match with the real robot measurements and the colony density in the local area. In detail, the nutrient or noxious environment condition is described by the formula:

$$f_1(\overrightarrow{m}_{p_{j,t}}, \overrightarrow{m}_{r_t}) = \frac{1}{M} \sum_{i=1}^{M} e^{-\frac{\left(m_{i,p_{j,t}} - m_{i,r_t}\right)^2}{2\sigma^2}} \tag{4.10}$$

where σ is tuned coherently with the robot measure confidence intervals.
The colony density is defined as:

$$f_2(p_j, P) = min \left\{ 1, \frac{1}{\nu N} \sum_{i=1}^{N} \left(e^{-\frac{\|p_j - p_i\|^2}{2\sigma^2}} \right) \right\} \tag{4.11}$$

where $\| \cdot \|$ is the Euclidean distance between two points, with $\nu \in [0, 1]$ and σ controlling the maximum colony size and the spatial radius respectively.

If a bacterium belonging to a colony C_i in a determined spatial radius is considered as an individual in a species S_i (colony), the corresponding deterministic differential equation which holds for large populations is:

$$\frac{dS_i}{dt} = f_1(S_i) \left(1 - \left(f_2(S_i, N) + \sum_{k \neq i} f_2(S_k, N) \right) \right) S_i \tag{4.12}$$

Note that if $f_2(S_i)$ is approximated with $\frac{S_i}{N}$, the logistic law is obtained while the growth is limited by the density and the size of the other colonies, with $\sum_k S_k \leq N$ as a boundary condition. Here it is assumed that a colony is determined by a small radius in which nutrient conditions and density can be considered constant. If a bacterium reproduces out of this radius, then it is considered either migrating to another colony or forming a new one. The spatial reproduction of a bacterium p depends on the environmental condition: if favorable, the bacterium reproduces in a small neighborhood; otherwise it migrates according to a normal distribution, whose variance is inversely proportional to the nutriment conditions.

$$reproduction(p) = \begin{cases} x_p = \mathcal{N}(x_p, \frac{\sigma_1}{f_1(p)}) \\ y_p = \mathcal{N}(y_p, \frac{\sigma_1}{f_1(p)}) \\ \vartheta_p = \mathcal{N}(\vartheta_p, \frac{\sigma_2}{f_1(p)}) \end{cases} \tag{4.13}$$

Note that as $f_1(p)$ approaches zero, the normal distribution tends to the uniform one: the bacterium is randomly dragged, wandering for attractive areas and the bet-hedging strategy is achieved.

This behavior turns out to be very effective, in particular when a robot is already roughly localized and a kidnap event occurs. In this case, as soon as the hypothesis measurements no longer match the real one, the reproduction rules at the base of the BCGA will provide an immediate response to the kidnap. That is, colonies will start to expand in a Gaussian way with a standard deviation proportional to the matching criteria. Moreover, since a kidnap is a "drastic" event when compared to the most common sensor problems (such as the inability of a laser rangefinder to deal with glass walls), the standard deviation will approach zero more quickly, providing an automatic resampling of the search-space. In this way, a complete "reset" of the environmental conditions is achieved, enabling the algorithm to look in areas previously considered noxious as well. Indeed, this approach is far more innovative than the Monte Carlo Filter (MCF), as no additional heuristic is required to "sense" the kidnap event since it is automatically handled by the dynamics of the equations.

4.2.2 Foreground Level: Multi-Hypothesis Choice and Interpretation

The competitive logistic model presented in the last section and its implementation within the BCGA represent a simple but flexible model for multi-hypothesis. Depending on the problem issues, a set of more complex equations and corresponding behaviors can be devised, as shown below in some practical examples. In the global localization problem, it is often the case that the hypothesis choice strategy is directly related to the algorithm. Strategies might include maximum probability, maximum fitness, et cetera. In a wider context, referring to sensor fusion, the multi-hypothesis characterization and its interpretation can be divided and independently carried out. More specifically, for any general problem setting, two possible strategies can be devised:

- Augment the complexity of species evolution model and keep a naive decision strategy.
- Keep the species evolution model simple and design a set of more accurate decision strategies using the distributions resulting from a simple BCGA.

Augmenting the model complexity requires a deeper investigation of the robot dynamics and behaviors related to the environment and the sensor measurements. Conversely, while a more simplistic model might be less robust in carrying on the multi-hypothesis, an accurate foreground strategy could compensate for this shortcoming. It is worth noting that the decision to modify the reproduction equation or the hypothesis choice depends highly on the experiment scenario. If the robot measurements are reliable, a naive reproduction scheme may be sufficient when combined with an accurate kinematic model. If the measurements are not sufficiently reliable, as in the case of laser sensors striking glass or when complex robot movements lead to phenomena such as sliding, the policy must be further investigated.

Choosing the best hypothesis is a good example of naive foreground strategy. In the case of the BCGA, the densest colony within the most nutrient area is selected. Unfortunately, this solution can lead to unrealistic optimum fluctuations. A more robust technique is achieved by introducing a weighted mobile temporal mean of the most likely hypotheses. If the aim is to preserve all plausible hypotheses, a proper multi-tracking strategy might be considered, e.g., performing a clusterization over the colonies and describing the trajectory of each hypothesis by the barycentre of a cluster. Alternatively, the bacterial reproduction schema might be modified when sensor data are known to be unreliable.

In this work, three different policies have been investigated:

- *Best colony* for simulation environment.
- *Mobile temporal mean* for real environment.
- *Modified reproduction* for real environment.

The mobile temporal mean has been adopted as a good compromise between efficacy and simplicity. Note that even though a similarity with the weighted resampling step of the MCF might be found when using the mobile temporal mean as a foreground strategy, a fundamental difference arises. In the MCF, weights affect the survival of hypotheses (particles), while in the BCGA, weights are introduced only to perform a comparison among colonies.

The modified reproduction is instead designed to augment the robustness of the hypotheses survival against measurement faults. The fitness reproduction chances of a bacterium do not depend only on fitness f_1 and density f_2, but also on ancestoral characteristics. The idea is that a bacterium, when reproducing, transmits its genes to the progeny and determines if they will be more or less effective in reproduction during following generations. A way to express this with formulae is:

$$r = f_{1_k}(1 - f_{2_k}) \tag{4.14}$$

$$h = \frac{1}{K}\sum_{i=1}^{k-1} f_{1_i} \tag{4.15}$$

$$\hat{r} = r + h - rh \tag{4.16}$$

where r is the reproduction probability previously introduced, h is the "genetic help" (equal to the average fitness of the ancestors over the generations), and $\hat{r}$ is the modified reproduction probability. This reproduction schema provides a better estimation of the hypotheses distribution, allowing for a simple foreground strategy such as the naive best-colony strategy.

4.2.3 Collaborative Localization

The BCGA algorithm provides an effective localization strategy when the single-robot context is considered. However, it does not take into account collaboration among robots. An algorithmic extension (CBCG) can be devised in order to exploit information derived by collaboration among robots. In detail, once two real robots (say r_1 and r_2) meet, i.e, they are within their range of visibility V, sensor data along with relative distance and orientation are exchanged. This information will be used to improve the localization process as follows (supposing two robots are moving in the environment and omitting the temporal index):

- Consider two robot populations P_1 and P_2, composed by single hypotheses $p_i^1 \in P_1$ and $p_j^2 \in P_2$, and let the real robot r_1 be seen by r_2.
- The robot r_2 communicates to the robot r_1 its relative distance and orientation along with a subset of its population for which $f_1(p_j^2) \geq \phi$ (where ϕ can be set arbitrarily) holds.
- Each p_i^1 exploits this information to refine areas potentially nutrient with regard to its own estimate. Specifically, p_i^1 updates its fitness through a modified version of the (4.10) able to take into account data sent by r_2.

The modified version of the nutrient or noxious environmental condition, for each $p_i^1 \in P_1$, is given by:

$$\hat{f}_1(p_i^1) = f_1(p_i^1) + f_3(p_i^1) - f_1(p_i^1) \cdot f_3(p_i^1) \tag{4.17}$$

where $f_1(p_i^1) = f_1(\overrightarrow{m}_{p_i^1}, \overrightarrow{m}_{r_1})$ (time index is not shown) and $f_3(p_i^1)$ represents the weighted projection of the estimate of r_2 on r_1, which can be defined as:

$$f_3(p_i^1) = max\left\{ f_1(p_j^2) \cdot e^{-\frac{\|p_j^2 - p_i^{1\to2}\|^2}{2\sigma^2}} \right\}, \quad \forall p_i^1 \in P_1, \forall p_j^2 \in P_2 \tag{4.18}$$

where $p_i^{1\to2}$ is the estimate of robot r_2 with respect to the hypothesis p_i^1 of the robot r_1. Furthermore, (4.17) can be viewed, from a probabilistic perspective, as a "union" that redefines a probability distribution with regard to additional information coming from another source.

4.2.4 Parameter Optimization

An open problem for the MCF, the BCGA and related techniques is the parameter optimization, such as the choice of the initial number of particles (or bacteria) or the definition of the variance for reproduction area. An *a priori* determination of these parameters is difficult. It can depend on the size of the deployment area, the ambiguity of both paths and sensors, as well as the kinematic model reliability. If the real robot path is available, the algorithm can be run several times with different parameter configurations and the resulting tracking errors can be compared through statistical tests. This way, parameters can be optimized and a satisfying performance can be achieved, lowering the number of bacteria.

In this study a non-parametric Wilcoxon rank-sum test [157] was adopted to compare median error vectors on the iteration steps. The Wilcoxon rank-sum test is a non-parametric statistical analysis of the differences in the distributions of two groups. This test is the equivalent of the Student's t-test for normal distributions, but relaxes the Gaussian requirement and allows for comparison through median and rank.

The test can be useful in performance comparisons when two robust indicators are derived from experimental settings. For this study, a performance indicator vector was generated with the aim of measuring the model behavior over time. Specifically, at each time step, the median (or mean) of the pose error resulting from 50 independent model runs is taken. In this way, a non-parametric distribution of pose errors over time is achieved. If two models have to be compared, e.g. the BCGA against the MCF or two BCGA with different parameter settings, the corresponding performance indicator vectors are calculated and compared with the rank sum statistics. Thereby, probability values explaining the model differences are obtained.

4.2.5 Computational Complexity

To evaluate the ability of an algorithm to run in an on-line context, the computational complexity becomes a very useful analysis. Therefore, a detailed theoretical study has been performed along with an empirical validation of the obtained results. The algorithm presents two nested loops in which the dominant operation is the density estimation function, linear with the population size. At first glance, the complexity turns out to be $\mathcal{O}(n^3)$ in the worst case, $\mathcal{O}(n^2)$ in the best case, and $\mu \cdot n^3 = \mathcal{O}(n^3)$ in the mean case, where $\mu \in [0, 1]$ is the mean reproduction factor. However, the density function can be dynamically calculated with increasing complexity, linear with the first loop, thus the worst case can be reduced by a factor of two and the mean case becomes $\mathcal{O}(n^3) = \frac{\mu \cdot n^2 \cdot (n+1)}{2}$. The complexity remains cubic, but with low constants. Coupled with the fact that in general the BCGA needs a lower number of bacteria compared to the number of particles needed for the MCF, its use in an on-line context is favorable. The next step will be an even lower bounded implementation.

4.3 Performance Evaluation

The proposed Bacterial Colony Growth Algorithm has been thoroughly investigated in both a simulated environment and with real robot data. Experiments carried out with the robot showed the capability of the algorithm to solve the localization problem in different real-world contexts, while simulations were fundamental for tuning the algorithm parameters, exploring the kidnap and validating the multi-robot collaborative strategy. The exposition is organized in two paragraphs, the first analyzes the behavior of the algorithm for single robot localization while the second reports some results for multi-robot.

4.3.1 Problem Settings

Computer Simulations. Simulations have been carried out in a framework developed under Matlab by the authors. This framework provides different kinematic models for the robot, such as the unicycle model, as well as an emulation for several sensors such as a laser rangefinder. Moreover, the environment is described by a set $\mathcal{M}$ of segments. This framework supports both a complete simulated context as well a test-bed to run data coming from a real robot. These two different operative modalities turn out to be very useful, both to test the correctness and the effectiveness of the algorithm. Details about robot, sensors and environment modeling are provided in Appendix A.

Real Robot Context. Experiments have been carried out on the mobile platform ATRV-Jr manufactured by iRobot. It is a skid steering vehicle mainly designed to operate in outdoor environments. The ATRV-Jr has 4 wheels differentially driven by 2 DC motors: the motion is achieved by a differential thrust on the wheel pairs at the opposite sides. The mobile robot is equipped with 17 sonar rangefinders, a laser scanner (Sick LMS-220), an inertial platform (Crossbow DMU-6X), and a GPS receiver (Garmin GPS35-HVS). The sensory system is connected to the ATRV-Jr's on board PC (Pentium II, 350 MHz) running Linux, through serial ports on a Rockeport multiserial port card. The robot is delivered with a software development environment called MOBILITY, which provides full access to the software servers available on the mobile platform. Each server is assigned to control a specific hardware component (sensors and actuators). In this way all interfaces are reachable from the network exploiting a CORBA interface.

4.3.2 Single Robot

Simulation

The simulated environment was configured as a large indoor area with several ambiguous zones (rooms), with a few poses uniquely defined (Fig. 4.1). The robot was simulated moving along a fixed path for 300 steps (step interval at 1s). A kidnap condition at time $t = 100$ was added. The simulated laser sensors had a limit of 8m, while two random zero-mean artificial noise variables were added to the kinematic model and to the observation model respectively.

In order to assess the BCGA performances and robustness, a preliminary phase of parameter tuning was performed. Then, 50 independent test runs were carried out. The final BCGA, after tuning, was configured with an initial random population of 300 bacteria, a maximum colony fraction size $\nu = 0.3$, a colony radius $\sigma_r = 10m$, a sensor measure deviation of $\sigma_m = 0.1$, a tolerance in pose of $\sigma_x^2 = \sigma_y^2 = 0.1$ and $\sigma_\vartheta^2 = 0.05$. The strategy for the best hypothesis selection was the best bacterium in the densest colony. This is a naive strategy for the hypothesis choice (in Section 4.3.2 a set of more effective strategies are presented) but it turned out to be satisfactory in this context.

According to the simulation results, the BCGA algorithm was able to carry on the multi-hypothesis and successfully localize the robot after a few iterations. It was also able to quickly re-localize the robot when a kidnap occurred (Fig. 4.11).

Robot in Real Environment

The ATRV-Jr was put in three indoor office environments:

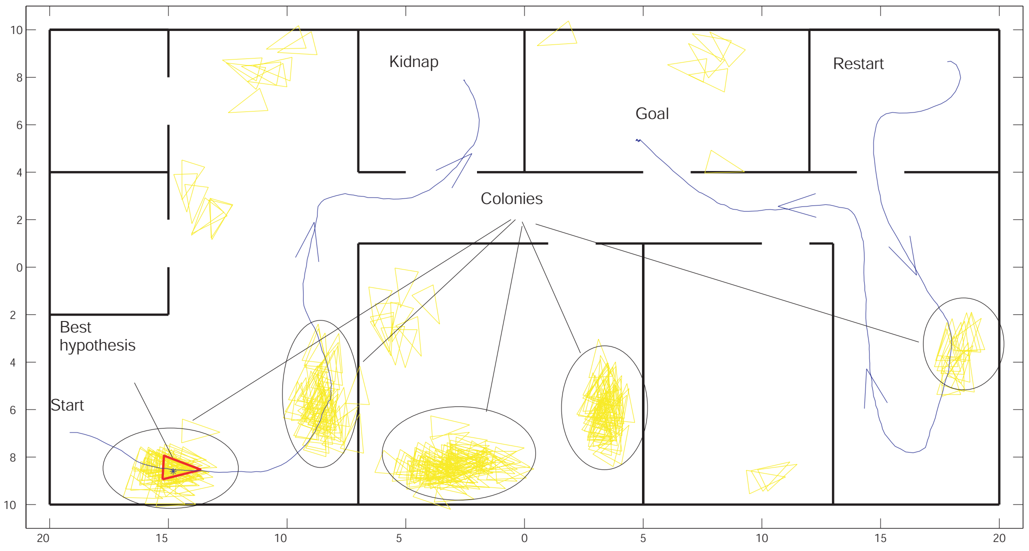

Figure 4.1: Simulated environment: multi-hypothesis keeping.

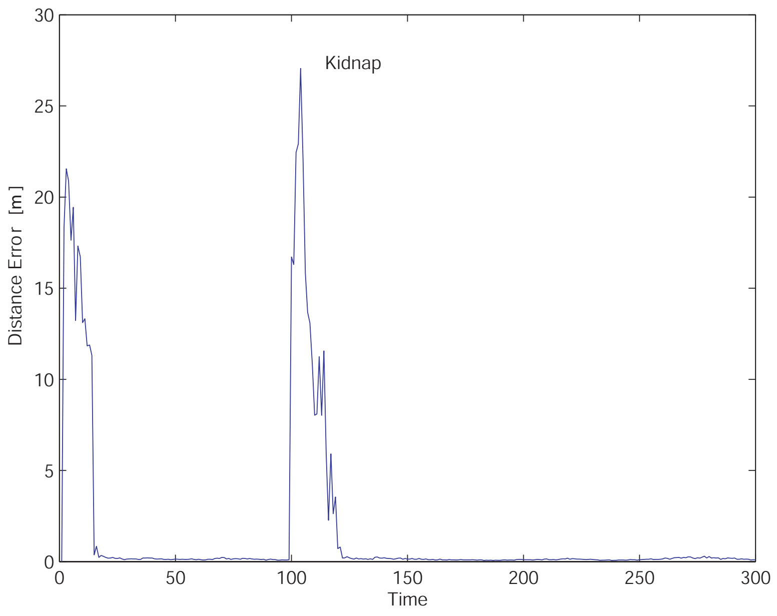

Figure 4.2: Simulation environment. BCGA: Median pose error over 50 trials. Naive best hypothesis choice.

- Corridor.
- Lobby.
- Entire building floor.

The environments were selected with increasing complexity and size. All of them contained ambiguous areas (including corridors, similar rooms, et cetera) and places in which both sensors and kinematics fail (glass doors, smooth floors, et cetera). Laser rangefinders were set to high definition and small range mode (8m), so that the overall coverage of the environment would not always be guaranteed.

The Corridor. The robot moved through the corridor, making 180° U-turns at each dead-end (Fig. 4.3). The sampling frequency was 5Hz and an accurate Kalman path estimation was available for comparison. The environment featured highly ambiguous pathways and areas, especially in the middle of the corridor and in the two almost identical niches at each end. Tracking was further complicated due to sliding phenomena impacting encoder data and noise affecting laser measurements (especially in the U-turn, where glass doors were also present). The *foreground* strategy was not limited to the trivial best-colony (or best-particle) choice. More complex problem settings demanded more robust hypotheses discrimination. Experimental results suggested that the simple competitive-logistic model was powerful enough to carry on the multi-hypothesis. However, a better tracking performance was obtained by exploiting the modified reproduction schema.

In particular, the following two configurations were taken into account:

1. Mobile temporal mean with the simple-competitive logistic law.
2. Modification of the reproduction law as in eq. (4.16).

In order to assess robustness and performance, a set of 50 independent runs were collected both for the first and the second strategies, with different parameter settings. For the mobile mean strategy, the optimally tuned BCGA was set up with the following parameters: an initial random population of 200 bacteria (as it was a smaller area compared to the simulated one); a maximum colony fraction size $\nu = 0.5$; colony radius $\sigma_r = 1m$; sensor measure deviation of $\sigma_m = 0.1$; and tolerance in pose of $\sigma_x^2 = \sigma_y^2 = 0.05$ and $\sigma_\vartheta^2 = 0.005$. Colonies grew and moved coherently with the robot poses, except in the region corresponding exactly to the U-turn. Here, the longest lasting colony (and so far, the correct one) depleted (due to the inability of sensors to properly work in presence of elements made with glass and the imprecision of the kinematic model), but recovered rapidly.

Another difficulty occurred in the middle region of the corridor. Due to the symmetry of the environment, two high-density colonies were growing and moving in opposite directions. The depletion experienced during the U-turn, along with the similarity of sensor data readings, due perhaps to the limited laser range,

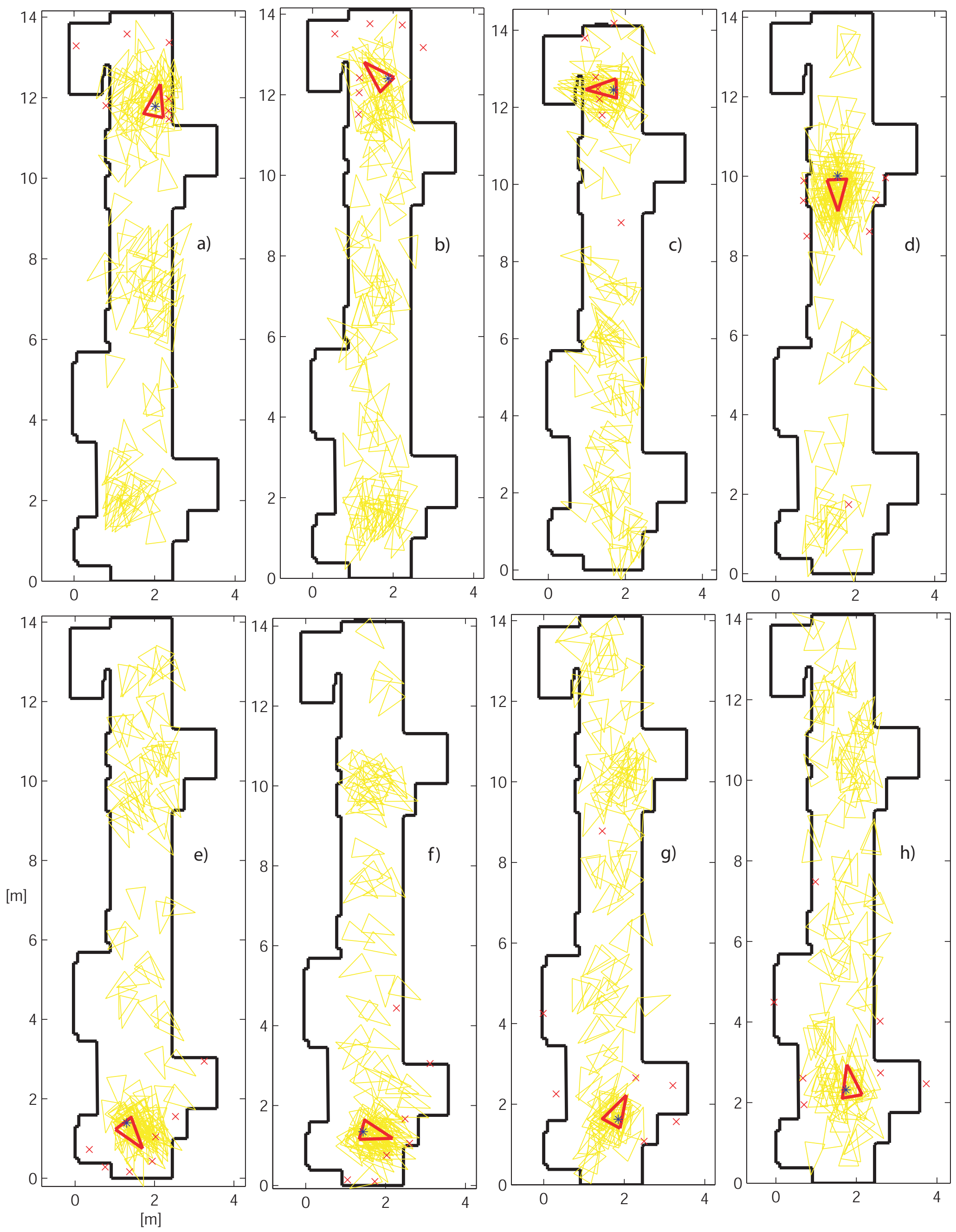

Figure 4.3: Real environment - Corridor - BCGA.

made the best mobile mean fail occasionally. Fig. 4.3 shows several steps of the algorithm's execution, where the thick (red) triangle represents the best hypothesis while the (blue) star is the real robot pose. Colonies are created w.r.t. the locations which better match with the sensor data, e.g. steps a, b, f. Colonies expanded (enhancing the state-space exploration) when in the presence of ambiguous areas, data or kinematic failures, e.g. steps g, h. Fig. 4.4 shows the median tracking error over 50 independent runs. Note that the mobile mean policy leads to a quick recovery from the U-turn depletion, even though problems in the middle corridor are experienced.

Experimental results indicate that the modified reproduction schema combined with the naive best-colony strategy performs better. In particular, a lower localization error is experienced and a reduced number of bacteria is required in order to successfully localize the robot (tests were made with 30 and 300 bacteria). Although a better performance is always experienced when the number of bacteria is increased (no matter what strategy is adopted), the second strategy outperforms the first even when considering only 30 bacteria. In this context, the Wilcoxon-test could be properly exploited to find out the optimal number of bacteria to use w.r.t. a desired error level. Fig. 4.4 and 4.5 show the algorithm performance when considering the mobile mean and the modified reproduction schema with the best-colony strategy.

The Lobby. This second environment (Fig. 4.6) presents a wider area when compared to the corridor previously discussed. Here, the robot started from the bottom and travelled upward, turning around and returning to the bottom again. The environment was less ambiguous, but the available map was less accurate as well, e.g. the slope of the incline on the top wall was incorrect. Again, 50 experiments were run and a Kalman path estimation was available for comparison.

The modified reproduction schema presented in eq. (4.16) was used and the number of bacteria was varied (30, 100 and 300). The rank-sum statistics again showed better performances for the 300-bacteria (which explains the lower recovery time after measurement failures, Fig. 4.7 and 4.8). In addition, the median error was below 0.5 meters for all settings. In the turning region, although the algorithm suffered from map inaccuracy, it was robust enough to track the robot (Fig. 4.7 and 4.8). Fig. 4.6 shows several steps of the algorithm execution. The thick (red) triangle represents the best hypothesis, while the (blue) star is the real robot pose. As for the previous experiment, colonies were created w.r.t. the locations which better matched the sensor data, e.g. steps b, c, d. Moreover, an expansion of colonies was noticed when in the presence of environmental ambiguities, e.g. steps a, f.

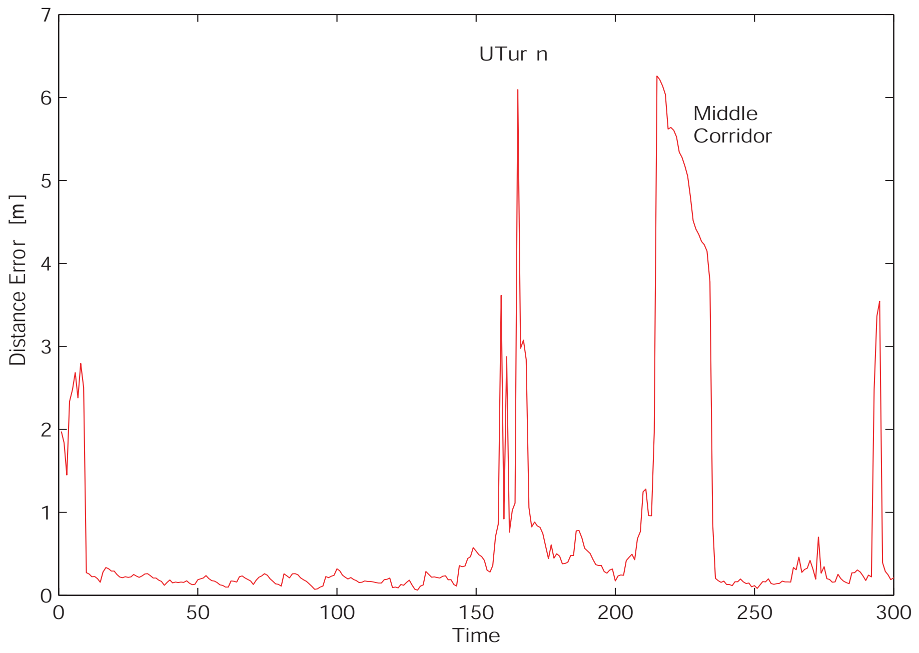

Figure 4.4: Real environment - Corridor - BCGA: Median pose error over 50 trials. Mobile temporal mean with the simple-competitive logistic law.

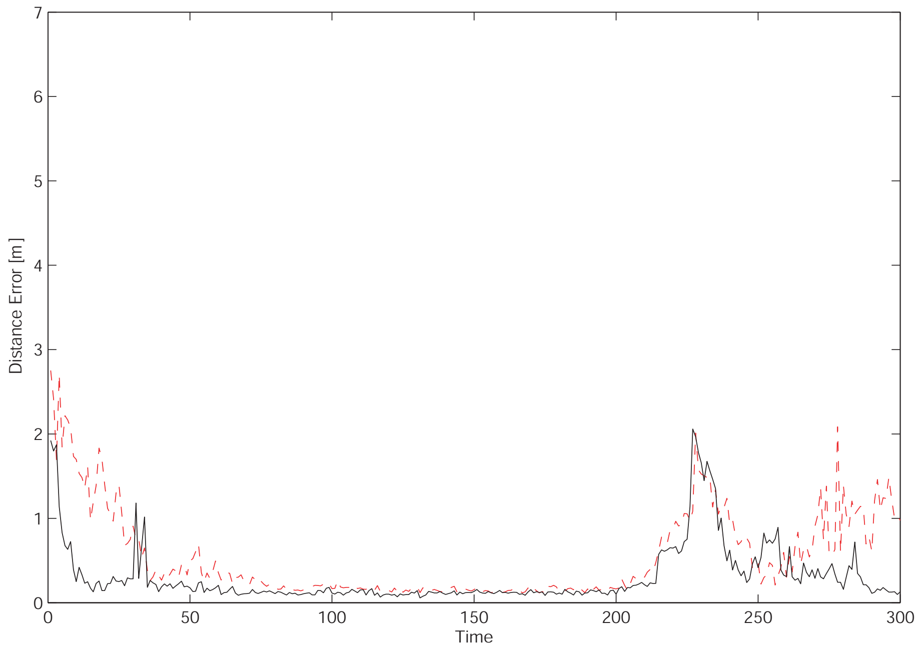

Figure 4.5: Real environment - Corridor - BCGA: Median pose error over 50 trials. Modified Reproduction Law with Naive best hypothesis choice. Runs with 300 (solid black line) and 30 (dash red line) bacteria.

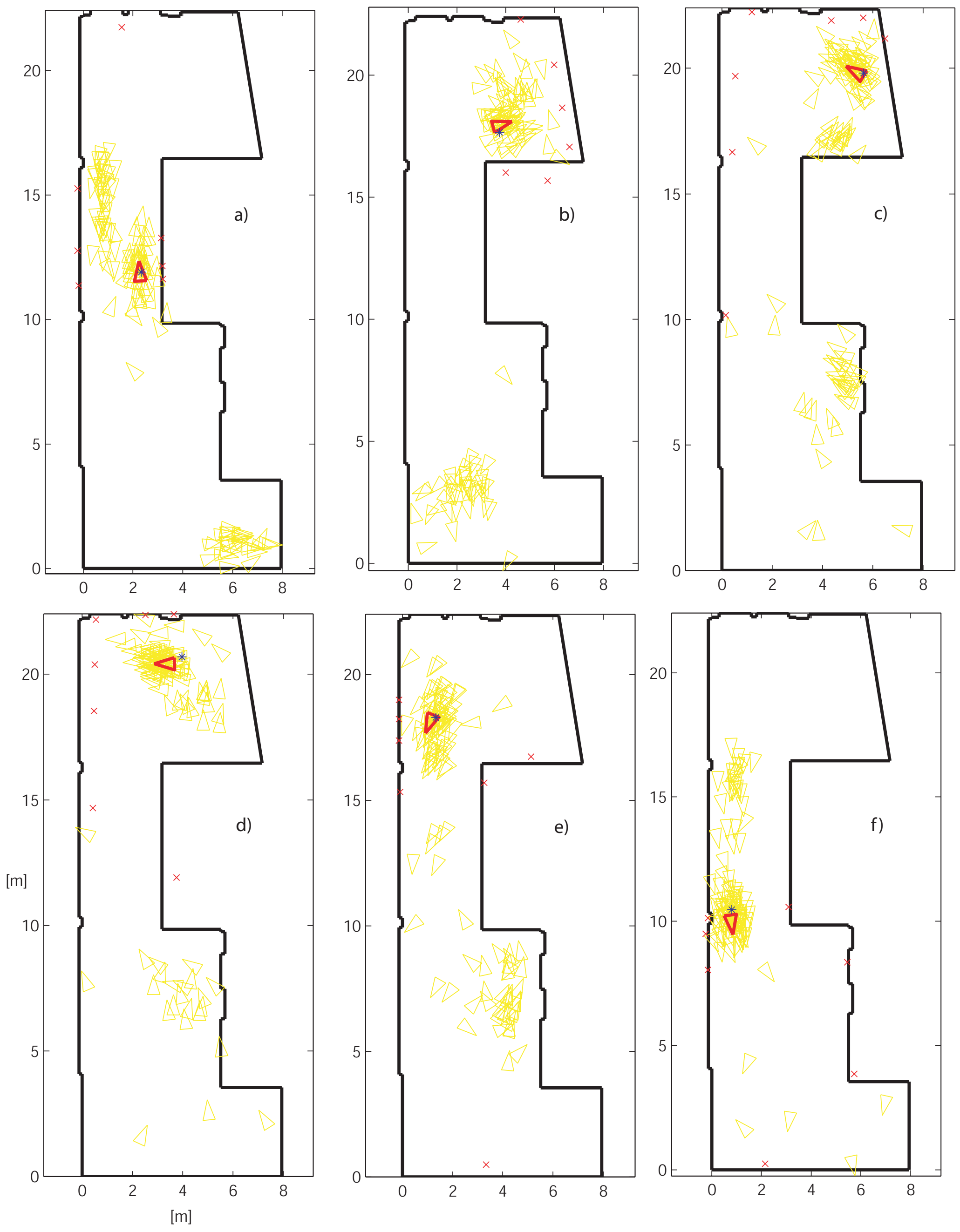

Figure 4.6: Real environment - Lobby - BCGA.

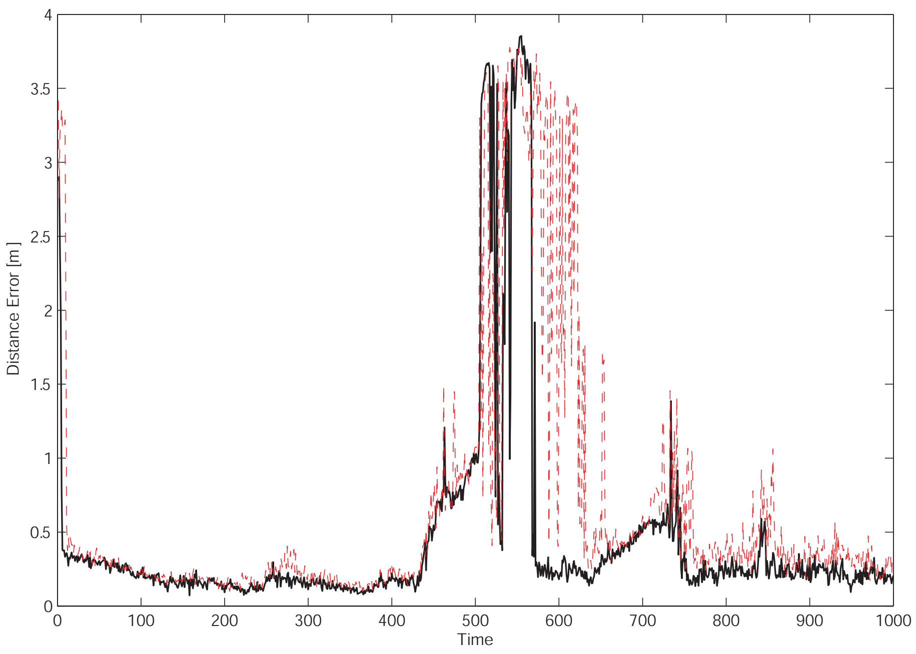

Figure 4.7: Real environment - Lobby - BCGA: Median pose error over 50 trials. Modified Reproduction Schema for best hypothesis choice. Runs with 300 (solid black line) and 100 (dash red line).

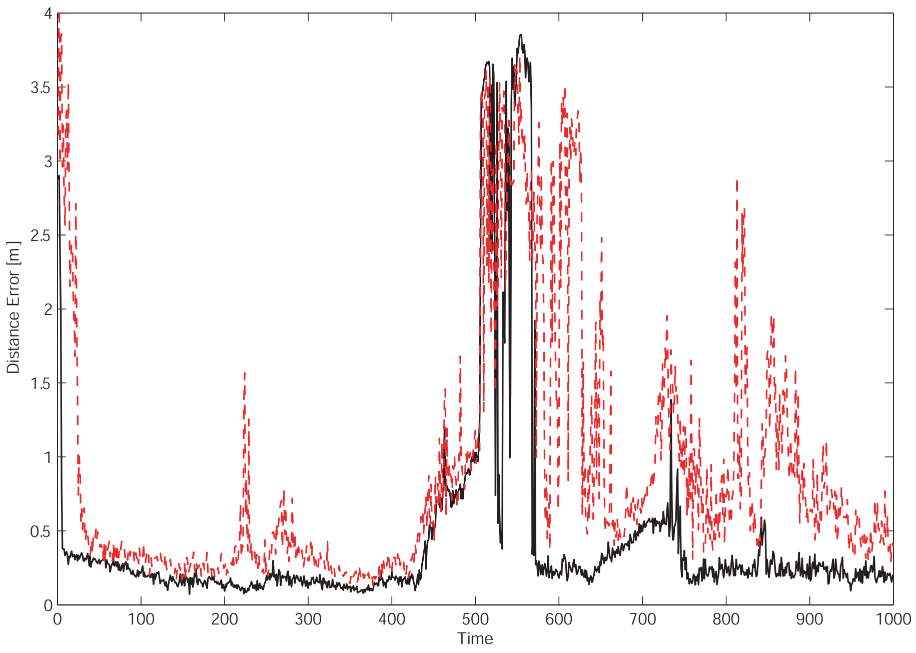

Figure 4.8: Real environment - Lobby - BCGA: Median pose error over 50 trials. Modified Reproduction Schema for best hypothesis choice. Runs with 300 (solid black line) and 30 (dash red line) bacteria.

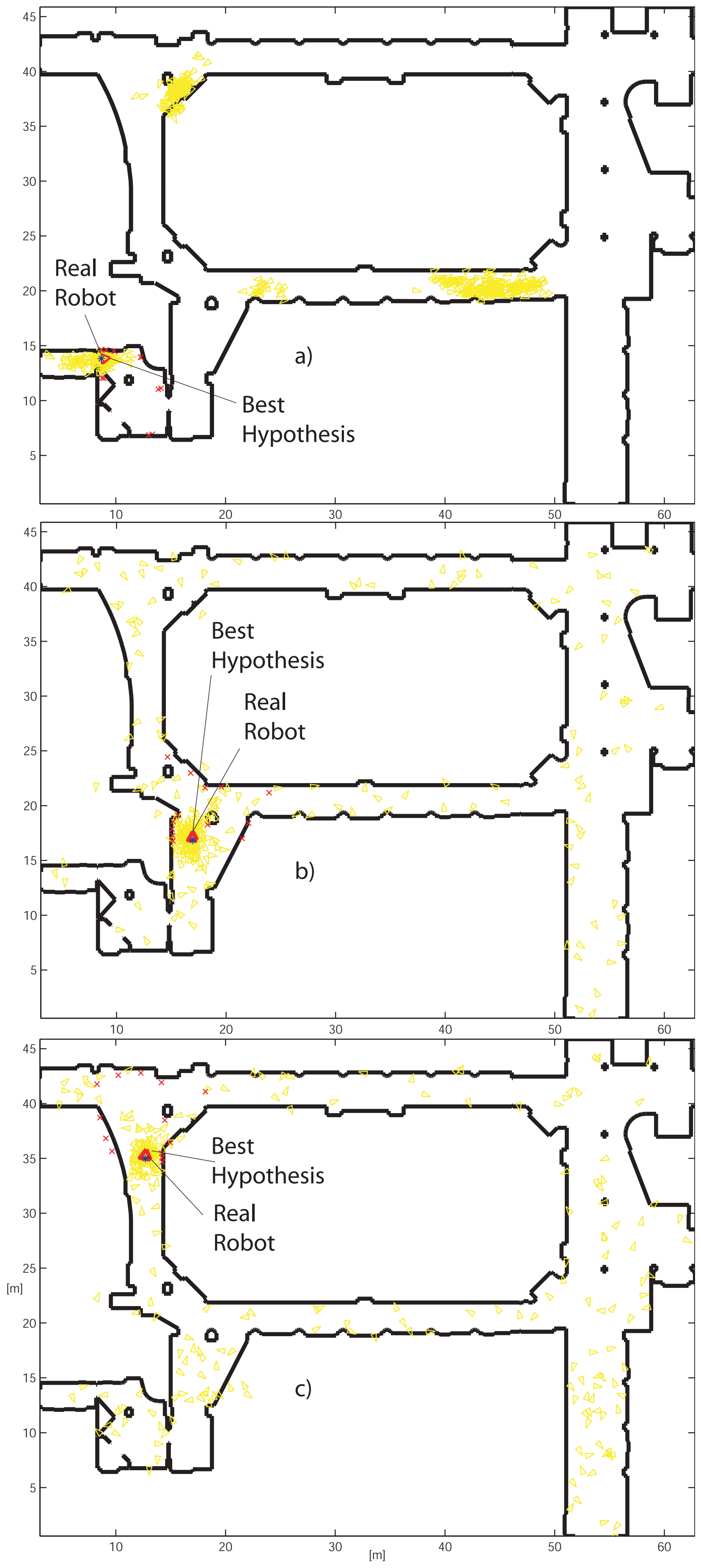

Figure 4.9: Real environment - Entire Building Floor - BCGA.

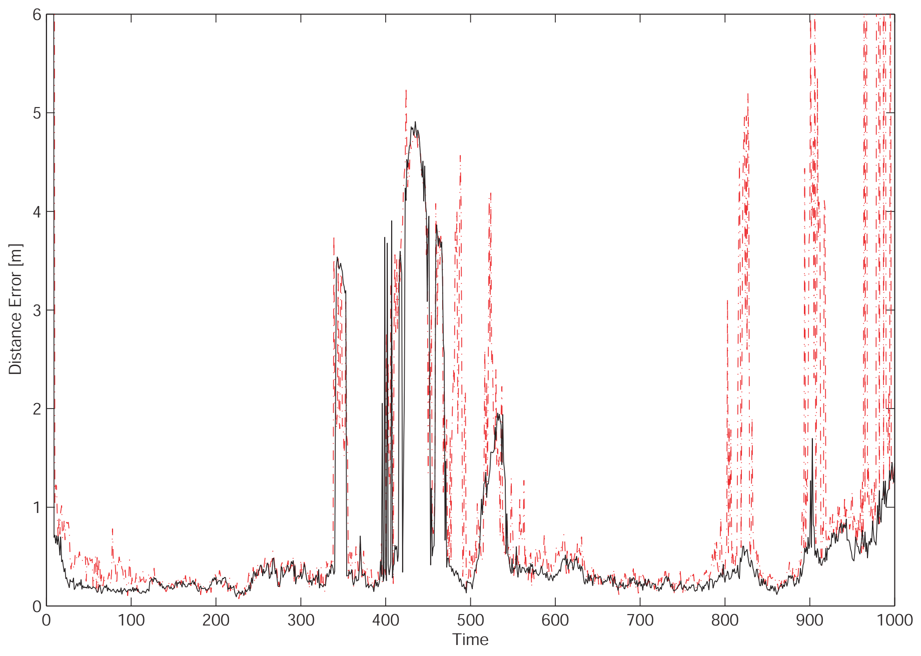

Figure 4.10: Real environment - Entire Building Floor - BCGA: Median pose error over 50 trials. Modified Reproduction Schema for best hypothesis choice. Runs with respectively 300 (solid black line) and 100 (dash red line) bacteria.

Entire Building Floor. This is the largest environment where the BCGA was tested. It is the first floor of the Computer Science Engineering Dept. of Roma TRE University. It features a smooth, glossy ceramic floor, rough white walls and several glass doors and windows. The robot started from the bottom-left small niche and moved towards the first large area with the pillar surrounded by glass doors. Then, it continued through the left corridor and finally turned right into the upper horizontal corridor (Fig. 4.9). The total path was 1000 time steps, with sampling frequency at 5Hz. Also in this case, a reliable Kalman path estimation was available to evaluate the algorithm tracking capability.

The BCGA was run 50 times with three different population sizes (300-150-50). With less than 150 bacteria it was almost impossible to find and track the robot, while with 300 bacteria the pose error was acceptable. The modified reproduction strategy turned out to be the only one able to provide good performance, even though some problems were experienced, in particular, along the last part of the path (the long corridor). Fig. 4.10 shows the median errors w.r.t. a population size of respectively 300 (solid black line) and 150 (dash red line) bacteria.

4.3.3 Multi-Robot

In order to assess model performances, a set of 100 independent BCGA and CBCG runs was executed, recording the pose error at each time step, designing and maintaining a fixed path for two robots that would allow also mutual localization. Table 4.1 describes the algorithm parameters setting adopted for simulations. Specifically, at each iteration of a given trial, a pose error was computed with respect to the best hypothesis choice described above. Median and mean errors over the 100 trials were evaluated as performance indicators, obtaining a non-parametric distribution of the errors over time to be statistically investigated.

Table 4.1: Algorithm Parameters Setting

Parameter	Description	Value
N	Population Size	$50 \div 100$
L	No. of Pattern Beams	7
l	Beam Range [m]	12
σ_l	St. Dev. Laser Beam Noise [cm]	10
σ_p	St. Dev. Spatial Radius Bound [m]	2
ν	Density Bound [%]	50
σ_1	St. Dev. x-y Reproduction Area [cm]	10
σ_2	St. Dev. Angle Reproduction Area [rad]	0.05
V	Range of Robot's Visibilty [m]	15

Two environment, specifically designed to make convergence challenging, are proposed. In detail, Fig. 4.11 and 4.14 show such environments along with the related paths for two robots. The first environment presents rooms with structural similarity among them, while the second allows for symmetric paths. Thus ambiguous situations can arise in both cases. Note that, for clarity of exposition only simulations involving two robots are provided. However, collaboration it is not limited to that, indeed it scales well with respect to the number of robots.

Figs. 4.12 to 4.16 show the performances of the algorithm for both robots in the two environments. Dashed lines (red) describe localization errors for the autonomous policy, while solid lines (blue) refer to the collaborative policy. In both scenarios communication among robots is available only in the second half path: thus no improvement can be provided from the collaborative approach until this event occurs.

The median, which cuts off outliers, underlines the accuracy of the localization technique showing a negligible error. Conversely, the mean reveals the robustness more broadly: the CBCG is more effective than the BCGA in terms of successful trials. This means that, in the ill situations for which the convergence lacks for one robot, processing the information from the mutual localization enhances the hypotheses convergence. Furthermore, a Wilcoxon rank test [157], detailed in Tab. 4.2, corroborates this result proving a significant statistical difference ($p \leq 1.2 \cdot 10^{-4}$) in performances when considering collaborative against autonomous strategy.

4.4 Considerations

In this chapter a new, biology-inspired robot localization approach has been proposed. The framework, the *Bacterial Colony Growth Algorithm*, is composed of two different levels of execution: a *background level* and a *foreground level*. The

Table 4.2: Wilcoxon rank test

Robots	p-value	rank sum	z-value
Environment 1			
R1	$4.5368 \cdot 10^{-7}$	2.6366	5.0449
R2	$8.2272 \cdot 10^{-9}$	$2.6906 \cdot 10^{4}$	5.7637
Environment 2			
R1	$5.1633 \cdot 10^{-7}$	$2.3072 \cdot 10^{4}$	5.0201
R2	$1.2119 \cdot 10^{-4}$	$2.2274 \cdot 10^{4}$	3.8437

first takes advantage of models of species reproduction to maintain the multi-hypothesis, while the second selects the best hypotheses according to an exchangeable specialized strategy, usually problem dependent. Indeed, this modular structure makes the algorithm very adaptive when considering different scenarios and objectives. The framework has been successively extended to deal with the multi-robot context by introducing the Collaborative Bacterial Colony Growth strategy (CBCG). Collaboration is exploited any time robots are within their range of visibility by exchanging sensory data along with their relative distance and orientation. Sender's information is exploited by the receiver to redefine areas that can be nutrient with regard to its estimate. This way, in the ill situations for which the convergence lacks for one robot, processing the information from the mutual localization enhances the hypotheses convergence.

Real experiments, carried out within three different environments with different characteristics and incremental difficulties, have been performed to validate the effectiveness of the proposed BCGA. Additional tracking strategies, more suitable for a real context, have been devised and discussed. Experimental results have shown the BCGA capability to maintain the multi-hypothesis in these scenarios. Moreover, thanks to the specialized foreground strategies, satisfactory tracking capabilities have been achieved. On the other hand, simulations along with proper statistical analysis have been run to validate the CBCG extension. According to the simulation results, the localization effectiveness increases drastically, in terms of robustness, when collaboration is properly exploited. This is corroborated also by the result of the Wilcoxon rank test. Indeed, it shows a significant statistical difference in performances when considering the collaborative strategy activated against the autonomous one.

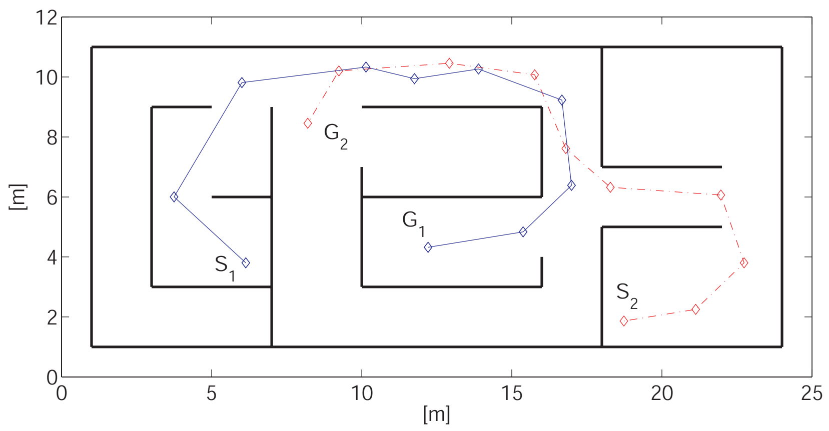

Figure 4.11: Environment 1: $S_{\{1,2\}}$ are start points, $G_{\{1,2\}}$ are goal points.

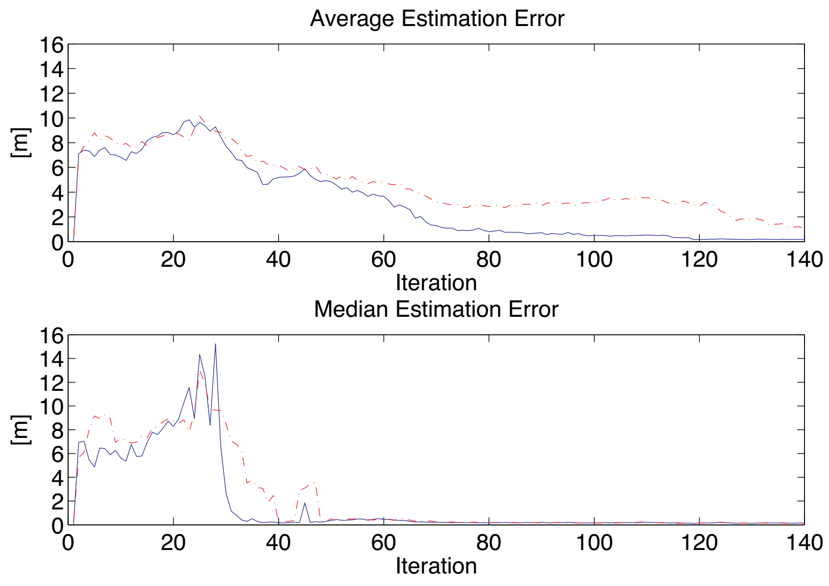

Figure 4.12: First Environment, Robot 1.

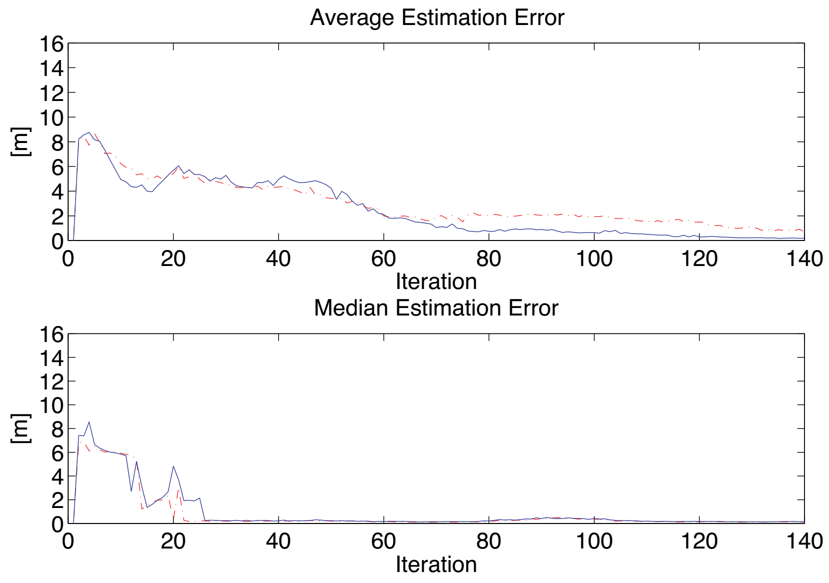

Figure 4.13: First Environment, Robot 2.

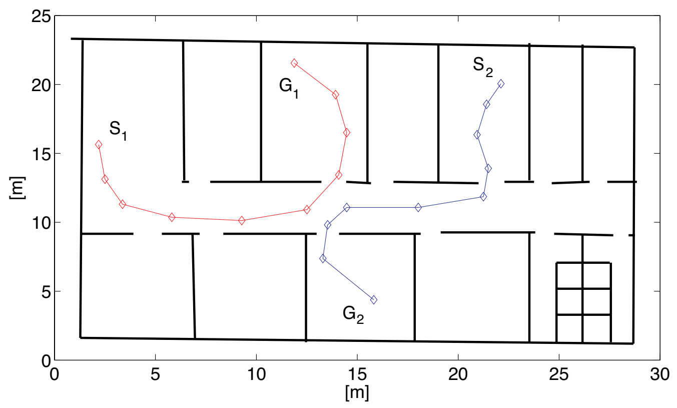

Figure 4.14: Environment 2: $S_{\{1,2\}}$ are start points, $G_{\{1,2\}}$ are goal points.

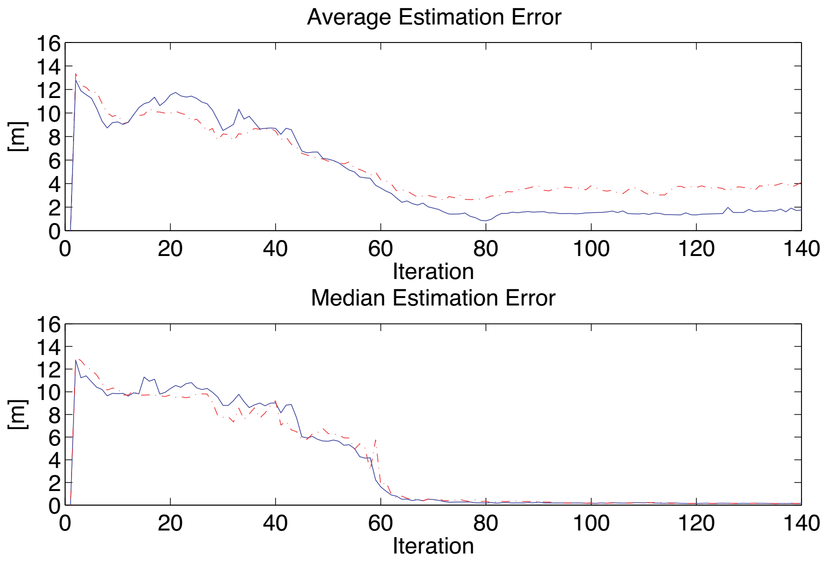

Figure 4.15: Second Environment, Robot 1.

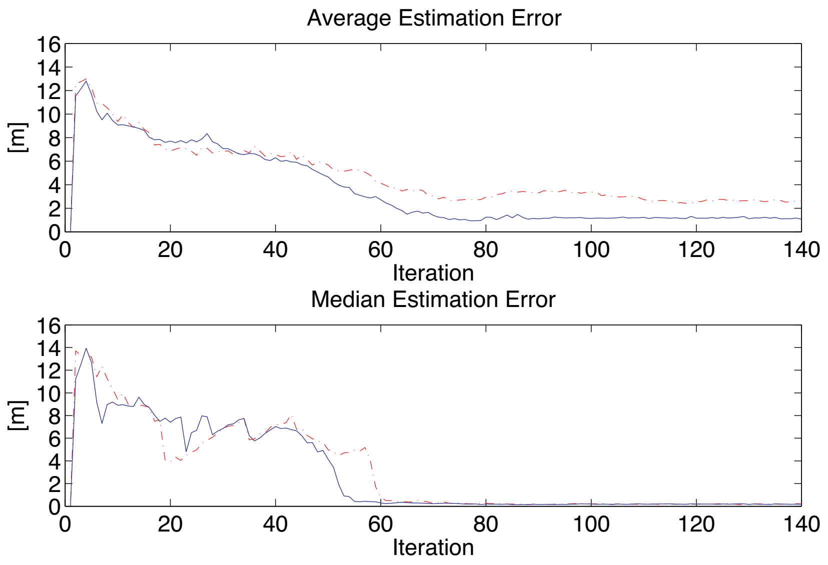

Figure 4.16: Second Environment, Robot 2.

Chapter 5

Algorithms Discussion and Comparison

Three different approaches for the Localization Problem in Mobile Robotics have been proposed thus far. In chapter 2 an enhanced Monte Carlo Filter aiming to overcome the classical MCF drawbacks has been proposed. In chapter 3 an evolutionary approach which provides a natural way to carry on the multi-hypothesis by exploiting the complex network theory for the deployment of the population is described. In chapter 4 an alternative biology-inspired framework for maintaining the multi-hypothesis is devised. Here, a comparative analysis is provided.

5.1 Algorithms Comparison

In the previous chapters three different approaches for the Localization Problem in Mobile Robotics have been described. Here, these approaches are compared to give evidence of their advantages and weaknesses. For this reason, the way how the multi-hypothesis is kept over time and how the kidnapped robot problem is faced are briefly recalled for each algorithm. Successively, a comparative statistical analysis is described. In detail, two different aspects of interest are investigated: the accuracy of the estimation and the computational complexity.

5.1.1 Multi-hypothesis paradigm

One of the crucial aspects of the design of an algorithm for global localization in mobile robotics, is the capability to carry on the multi-hypothesis over time. Indeed, when a robot is moving within an environment characterized by symmetries, its localization module should keep track of all the likely locations until enough data is gathered to distinguish the correct hypothesis. This aspect was taken into account when developing all of the proposed algorithms.

For the CE-MCF this is achieved by means of a dynamic clusterization. Hypotheses are grouped by exploiting a density-based notion of cluster and the survival of different hypotheses is achieved by performing the reproduction locally within each cluster. At the same time, due to the capability of the clustering algorithm to collect *noisy* points, i.e., points that do not belong to any cluster, the diversity among particles is preserved.

For the SSGA this is achieved by exploiting the complex network theory for the spatial deployment of the population. In fact, the introduction of spatial structures in evolutionary algorithms helps to create evolutionary niches, while exploiting complex networks model such as the Watt-Strogatz, guarantees their persistence. Now, since niches describe regions where particular solutions are preserved, the maintenance of the multi-hypothesis is obtained.

For the BCGA this is achieved by exploiting the logistic models of species evolution. In particular, the framework is inspired by the observation that some families of bacteria tend to form colonies when in presence of favorable conditions. Bacteria representing hypothetical robot locations move according to the kinematics equations. However, their tendency to group and form colonies is regulated by the logistic equations where the favorable conditions are given by the measurement match and over-population control.

5.1.2 Kidnapped Robot problem

The kidnapped robot problem is probably the most captious among the three instances of the localization problem. In fact, starting from a well-localized robot, it implies both the capability to recognize a sensorial fault (kidnap detection) and re-locate the robot. In particular, being able to recognize a sensorial fault is not trivial at all. Indeed, the algorithm at the same time must be robust enough not to be deceived by a "noisy" situation, e.g. a temporary inability of the sensor to perceive the environment such as laser-beams against glass elements, and sensitive enough to figure out when something really is wrong.

For the CE-MCF the detection of a kidnap event is achieved by systematically checking the degeneracy of the population and performing a resampling step anytime this value becomes lower than a pre-defined threshold. This can be explained by observing that a drastic degeneracy is experienced in few steps anytime a kidnap occurs. As a consequence, a kidnap can be treated as a special case of the degeneracy problem. Note that, this does not hold for the classic MCF since the depletion is a severe problem on its own. On the contrary, the CE-MCF is able to preserve solutions and mitigates the degeneracy by exploiting the dynamic clusterization. Therefore, a drastic depletion is a good indicator of a kidnap event.

For the SSGA the detection of a kidnap is performed by a dedicated technique which exploits the fitness function along with the edge function. In particular, the former measures the compliance between real data (coming from the robot) and expected one (computed by each individual) while the latter evaluate the dispersion of the population. In this way a simple way to detect a kidnap is obtained since a high percentage variation of the fitness along with a considerable dispersion of the population (a high value of the edge function) are reliable symptoms of a kidnap.

For the BCGA a kidnap is automatically detected as a side-effect of the of the species evolution model at the basis of the reproduction, which is driven by the environmental condition. In particular, anytime the environment becomes noxious, bacteria migrate elsewhere according to a normal distribution whose standard deviation is inversely proportional to the nutriment condition of the environment. Now, since a kidnap drastically change the environmental conditions, the standard deviation will approach zero very quickly and therefore bacteria will be automatically spread over the whole environment.

5.1.3 Performance Analysis

The environment shown in Fig. 5.1 was exploited for the comparative analysis. It describes a portion of the Department of Computer Science and Automation of the University of "Roma Tre". It was chosen as a consequence of its particular structure. In fact, it is composed of two long corridors with several indentations which lead to various symmetrical areas difficult to distinguish. Therefore, it turns out to be a very tough test bed for the validation of a localization algorithm.

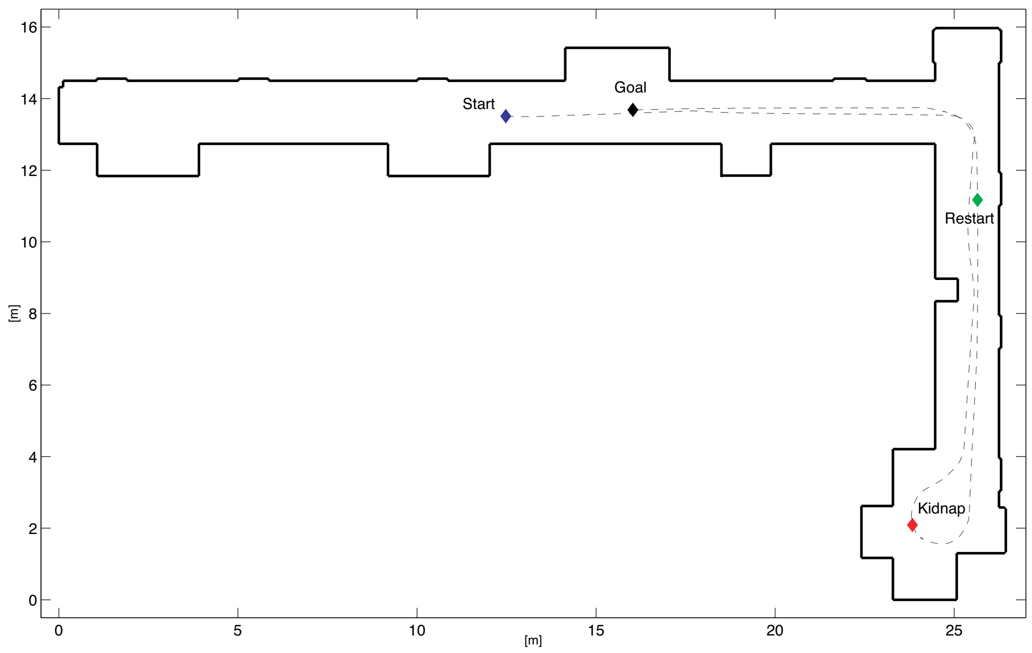

Figure 5.1: Test bed for comparative analysis.

In regard to the experiment, the ATRV-Jr robot started moving from the point *Start*. After around 480 iterations it was kidnapped from the point *Kidnap* and "tele-transported" to the point *Restart*. From there, it kept moving until the point *Goal* was reached. Each algorithm was run 30 times and a statistical analysis was performed. The following aspects of interest were investigated: capability to carry on the multi-hypothesis, to re-localize the robot after a kidnap and the accuracy of the estimation. A reliable estimation of the real path, computed by a Kalman Filter technique, was available for comparison.

Experimental results in terms of mean and median values over 30 trials are given in Figs. [5.2, 5.3], Figs. [5.4, 5.5], and Figs. [5.6, 5.7]. Respectively for the CE-MCF, for the SSGA and for the BCGA. Note that, a better insight of the algorithms performance is provided by exploiting both mean and median operators. In fact, while the median cuts off outliers by considering mainly the successful trials to analyze the accuracy of the localization technique, the mean reveals the robustness more broadly taking into account any outcome.

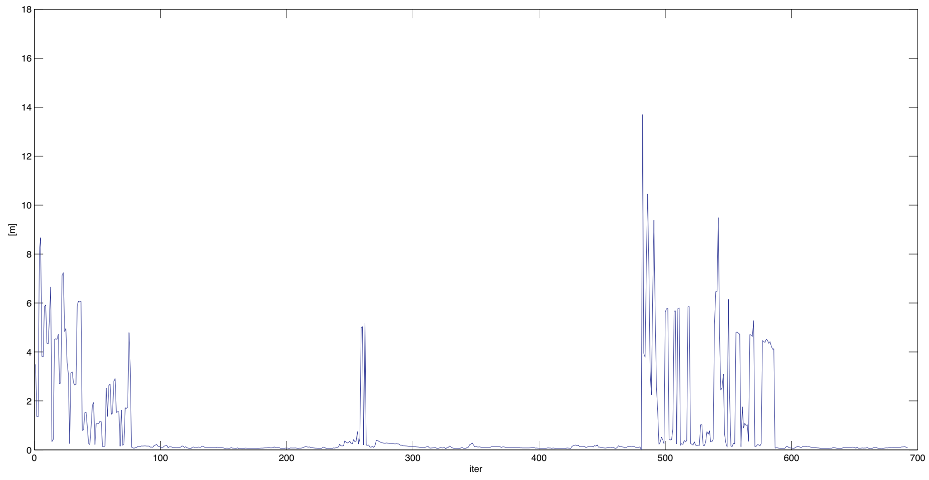

Figure 5.2: CE-MCF - 3000 particles - Mean

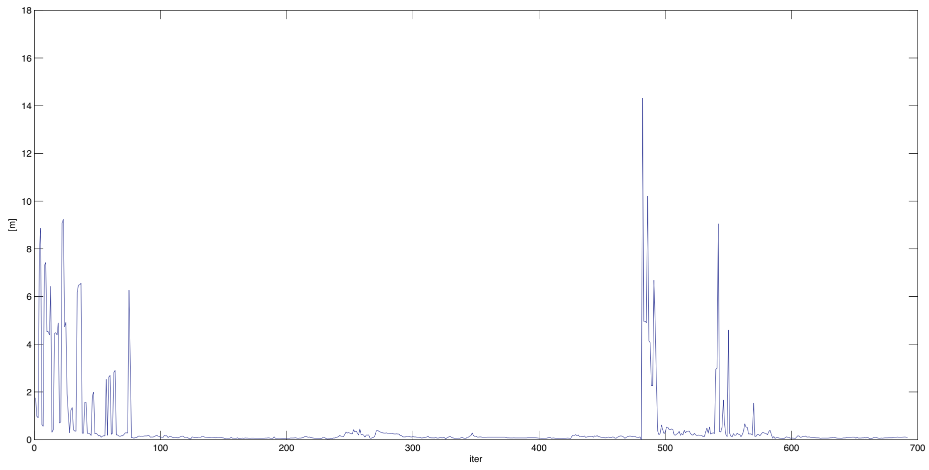

Figure 5.3: CE-MCF - 3000 particles - Median

According to the experimental results, the proposed algorithms were able to effectively solve the global localization problem along with the kidnapped robot problem. Note that, due to the presence of highly symmetrical areas within the environment, algorithms were sometimes temporarily deceived. However, the exact location was always re-established taking advantage of the capability to carry on the multi-hypothesis over time.

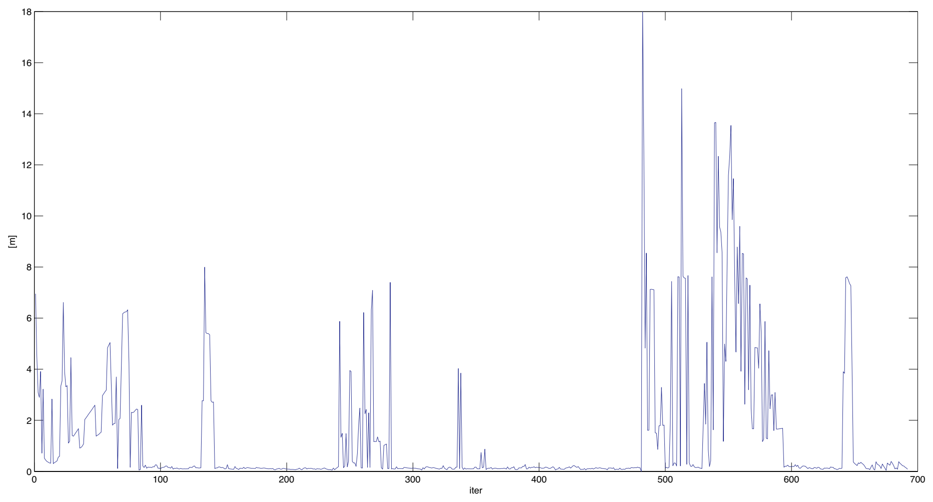

Figure 5.4: SSGA - 600 chromosomes - Mean

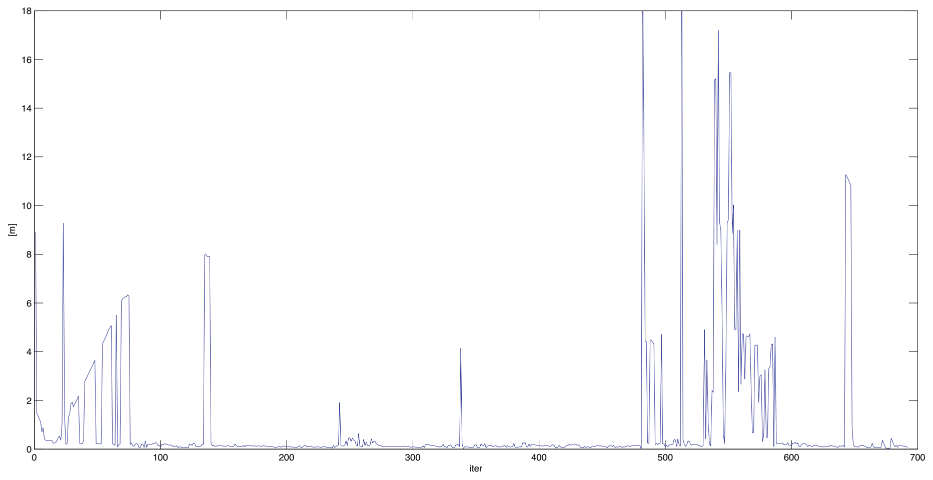

Figure 5.5: SSGA - 600 chromosomes - Median

Figs. [5.2, 5.3] show the performance of the CE-MCF when considering 3000 particles. Experiments were performed with a number of particles ranging from 1000 to 5000. However, satisfactory performance in terms of capability to localize the robot and solve the kidnapped robot problem were achieved with a sampling of at least 3000 particles. By observing the two plots, the CE-MCF does not seem to be significantly influenced by the presence of symmetrical areas. This is due to the fact that after a while the cluster associated to the real robot location has become predominant over the others. Although this seems a good property, it might turn out to be counterproductive as the capability of the algorithm to recognize a kidnap is lowered. This limitation is inherent to the nature of the Monte Carlo Filtering techniques where the weight of particles directly influence the survival of hypothesis. The introduction of an evolutionary action along with clusterization mitigates this weakness but it cannot be completely overcome.

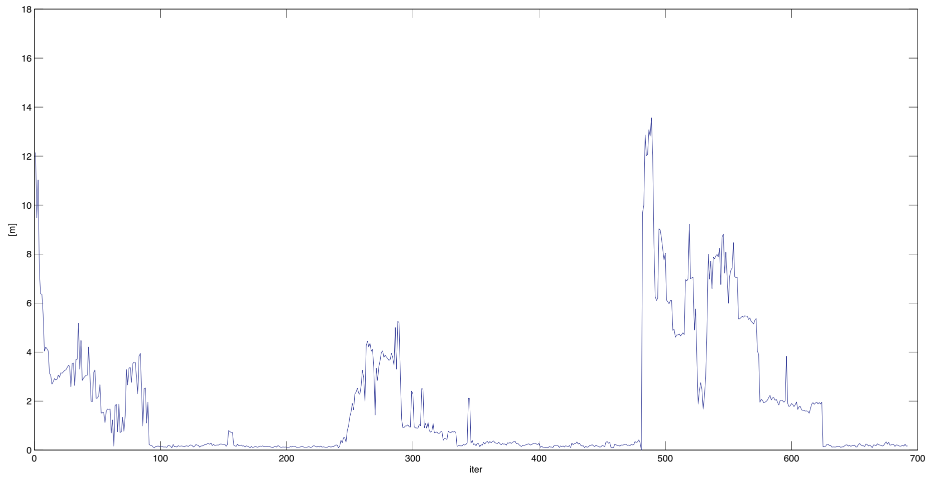

Figure 5.6: BCGA - 150 bacteria - Mean

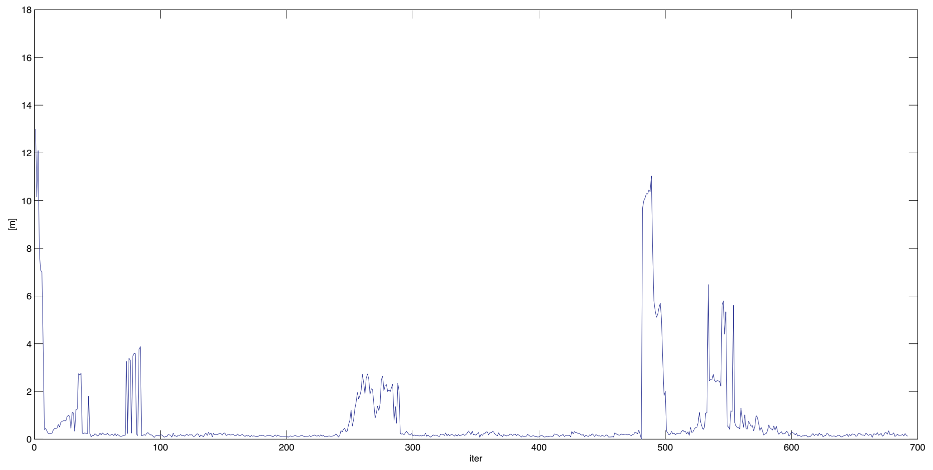

Figure 5.7: BCGA - 150 bacteria - Median

Figs. [5.4, 5.5] show the performance of the SSGA when considering 600 chromosomes. Experiments were performed with a number of chromosomes ranging from 300 to 1000. A satisfactory performance, in terms of ratio between accuracy of estimation and computational complexity was achieved considering at least a population of 600 chromosomes. Although some preliminary tunings of the kidnap detection function were required, the algorithm showed to properly localize the robot and solve the kidnapped robot problem. The presence of several peaks can be explained by the tendency of the algorithm to continuously "jump" from one niche to the others when data fits similar locations. In particular, the almost identical trend of both mean and median plots underlines the tendency of this approach to always explore the state space and therefore keep the multi-hypothesis over time.

Figs. [5.6, 5.7] show the performance of the BCGA when considering 150 bacteria. It is interesting to point out the capability of this approach to succeed with a very small population of bacteria. This is due to the introduction of the logistic models of species evolution which effectively carry on the multi-hypothesis over time. Moreover, the "jittering" phenomenon, experienced by the other two approaches when choosing the best hypothesis, is significantly mitigated by the use of the mobile temporal mean as a foreground strategy. Note that, this does not lead to any loss of diversity among bacteria as the multi-hypothesis is independently maintained over time by the background strategy.

5.1.4 Computational Complexity

The computational load is one of the most important aspects of an algorithm, particularly if it needs to run on-line. For this reason a comparative analysis of the computational complexity of the proposed algorithms is provided. In order to do that, the asymptotic notation (a mathematical notation used to describe the asymptotic behavior of functions) is considered. Its purpose is to characterize a function's behavior for very large (or very small) inputs in a simple but rigorous way that enables comparison to other functions [11].

Table 5.1: Computational Complexity

Algorithm	CE-MCF	SSGA	BCGA
Complexity	$\mathcal{O}(N \cdot M)$	$\mathcal{O}(k \cdot N \cdot M)$	$\mathcal{O}(N^3)$
Values	N=3000, M=80	k=3, N=600, M=80	N=150

Table 5.1 summarizes the computational load of the proposed algorithms along with the parameter setting used for the comparative analysis. In detail, in chapter 2 the computational complexity of the CE- MCF was derived to be $\mathcal{O}(N \cdot M)$, where N represents the number of particles involved and M is the number of segments required for the description of the environment. In chapter 3 the computational complexity of the SSGA was evaluated to be $\mathcal{O}(k \cdot N \cdot M) = \frac{(k-1) \cdot N \cdot M}{2}$, where k is the degree of each node (i.e. number of links for each node), N represents the number of individuals and M is the number of segments required for the description of the environment. Finally, in chapter 4 the computational complexity of the BCGA was determined to be $\mathcal{O}(N^3) = \frac{\mu \cdot N^2 \cdot (N+1)}{2}$ in the worst case, where N represents the number of bacteria and μ is the mean reproduc-

tion factor. In order to provide a fair comparison, the real parameters adopted for the previous experiment are considered. The number of segments required to described the environment was $M = 80$. The CE-MCF was configured with $N = 3000$ particles. The SSGA adopted the Watt-Strogatz model with $k = 3$ and a population of $N = 600$ chromosomes. Finally, the BCGA was setup with 150 bacteria.

According to this scenario, the SSGA followed by the CE-MCF, provides the lowest computational complexity while the BCGA has the highest complexity with the worst case being $\mathcal{O}(N^3)$. However, empirical evidence showed that the BCGA computational complexity is significantly lowered to $\mathcal{O}(N^2 \cdot logN)$ in the mean case. This observation coupled with the small number of bacteria required for solving the problem make the BCGA an appealing approach.

5.2 Considerations

In the previous chapters, three different approaches to the Localization Problem in Mobile Robotics have been proposed. In this chapter, the main features of these approaches have been reviewed and a comparative analysis has been performed. The capability to carry on the multi-hypothesis over time and solve the kidnapped robot problem have been mainly investigated. For this reason, an additional environment composed by two long corridors with several indentations has been exploited. In this way, the localization problem becomes even harder due to the presence of several structural similarities. Each algorithm was run 30 times and a comparative statical analysis was performed.

Experimental results, carried out with the robot ATRV-Jr manufactured by iRobot, showed comparable performances. Indeed, all the proposed algorithms were able to solve both the localization and kidnapped robot problems. Among them, the CE-MCF provided the more reliable path estimation, even though a considerable amount of particles was required to achieve such a result. The SSGA performed satisfactorily, even though a strong tendency to "jump" from one niche to the others was experienced when data fits more than one location. Finally, the BCGA turned out to be an interesting approach, although theoretically the worst case of the computational load is quite problematic. However, the limited number of bacteria required for solving the localization problem along with the empirical evidence of a significantly lowered mean case make this approach very appealing.

Future work should be devoted to experimentally validate the techniques proposed for the multi-robot scenario.

Part II

Localization In Sensor Networks

Chapter 6

Introduction

Wireless Sensor Networks (WSN) are at the forefront of emerging technologies due to the recent advances in Micro-Electro-Mechanical Systems (MEMS) technology. The inherent multi-disciplinary nature of WSN attracted scientists coming from different areas stemming from networking to robotics. WSN are considered to be unattended systems with applications ranging from environmental sensing, structural monitoring, and industrial process control to emergency response and mobile target tracking. Most of these applications require basic services such as self-localization or time-synchronization. The distributed nature and the limited hardware capabilities of WSN challenge the development of effective applications. Therefore, this field has drawn a great amount of attention within the research community in the last years.

6.1 The self-localization problem in Sensor Networks

A sensor network consists of a collection of nodes deployed in an environment that cooperate to perform a task. Each node, which is equipped with a radio transceiver, a micro-controller and a set of sensors, shares data to reach the common objective. Sensor networks provide a framework in which, exploiting the collaborative processing capabilities, several problems can be faced and solved in a new way. However, it comes along with several challenges such as limited processing, storage and communication capabilities as well as limited energy supply and bandwidth. Performing a partial computation locally on each node, and exploiting inter-node cooperation, is the ideal way to use sensor networks. Unfortunately, this *modus-operandi* is highly constrained by the reduced hardware capabilities as well as by the limited energy resources that makes communication very expensive in terms of life-time for a node. As a consequence, these constraints must be taken into account when developing algorithms able to operate in a distributed fashion.

Sensor networks can be of interest to different areas of application, ranging from environmental monitoring [42, 155], civil infrastructures [88, 108], medical care [142, 123] to home and office applications [144, 101]. In each field, the deployment of a sensor network has provided interesting advantages. For instance, in the context of environmental monitor the introduction of a sensor network made it possible to keep environments intrinsically threatening for human beings [155] under surveillance, or in the context of medical care it made it possible to remotely monitor the health condition of patients by continuously extracting clinical relevant information [123].

However, in order to build these application, some basic services, such as time synchronization or nodes localization, are generally required. In fact, basic middle ware services, such as routing, often rely on location information, e.g., geographic routing [29, 145, 94]. Specifically, the localization problem in Sensor Networks consists of finding out the locations of nodes in regards to any topology or metric of interest. This problem turns out to be difficult to solve. In fact in [84, 62] it was proven that a sufficient condition for a sensor network to be localizable cannot be easily identified. This holds even when considering the availability of perfect measurements. Further, several analyses showed that, especially when using the received signal strength indication (RSSI), having reliable ranging information is fairly practical [156, 158, 12].

6.2 State of the art

A taxonomy of localization algorithms for sensor networks can be drawn according to the computational organization, i.e., centralized and distributed, to the mechanism adopted for estimating location, i.e., range-based or range free, and finally in regards to the availability of anchors nodes, i.e. anchor-based or anchor-free.

Centralized algorithms exploit a central computer to perform all the complex computations using information gathered by nodes [54, 139, 30]. Distributed algorithms dispense the computation over the network, allowing each node to perform locally and compensating for the lack of global knowledge through an intensive collaborative processing [112, 45, 43]. Both schemes offer advantages and drawbacks. Centralized algorithms provide interesting performance but they lack in scalability and robustness. Distributed algorithms provide high robustness and scalability but the development of effective collaborative processing algorithms is challenging.

Range-based algorithms exploit point-to-point distances or angle estimates in order to perform the localization task [124, 141, 119]. Range-free algorithms do not make any assumption about the availability or reliability of this information [80, 102, 158]. Although range-free approaches are appealing as a cost-effective alternative to more expensive range-based approaches, their performance may lack in accuracy.

Anchor-based algorithms rely on the availability of location information for some special nodes in order to localize the network [63, 136]. Anchor-free methods determine the geometry of the network simply by exploiting local interaction among nodes [126, 159]. Anchor-based algorithms have the advantage of directly localizing nodes within a global reference frame, but their accuracy is affected by the number of anchor nodes and their distribution in the sensor field [35]. Conversely, anchor-free methods scale better and do not required expensive hardware, although only relative location estimates can be provided.

Centralized algorithms represent the first attempt to solve the localization problem in sensor networks. In [54], the authors propose the semi-definite programming approach (SDP) to solve the localization problem. The idea is to model geometric constraints between nodes as linear matrix inequalities (LMIs), then use the semi-definite programming theory to solve it. The result is a bounding region for each node, representing feasible locations where nodes are supposed to be. The idea to use a set of convex constraints in order to estimate the position of a node is very elegant, but it turns out to be inaccurate as constraints do not use precise data range. Moreover, the algorithm provides a good estimation only when having anchors densely deployed on the boundary of the sensor network, a condition that can not always be guaranteed. This approach is extended to deal with noisy distance measurements in [23]. The idea is to take advantage of an additional technique to mitigate the inaccuracy of the solution provided by the SDP formulation. In fact, the solution provided by the DSP, though not accurate, represents by the authors a good starting point for a gradient-descent method. Furthermore, numerical results show that by means of this approach it is possible to obtain a solution very close to the optimal one. This approach provides a significant improvement of the SDP-based algorithms' performance. However, the distributed formulation is the result of a clusterization and a local execution of the algorithm within each subset. Therefore, the computational complexity is merely mitigated reducing the number of nodes but the approach still remains almost centralized. In [139], the authors propose an algorithm that uses connectivity information, i.e., which nodes are within the communication range of which others, to derive the locations of the nodes in the network. This algorithm is based on multidimensional scaling (MDS), a set of data analysis techniques that display the structure of distance-like data as a geometrical picture [28]. It can be broken down into three steps. Starting with the given network connectivity information, an all-pairs shortest-path algorithm is run to roughly estimate the distance between each possible pair of nodes. After, the multidimensional-scaling is applied over these data to derive node locations. Finally, location estimates are normalized with respect to nodes whose position is known.

Distributed algorithms better fit the inherent collaborative nature of sensor networks. In [112], the authors developed an algorithm focused on providing more robust local maps. The idea is to split the problem into a sub-set of smaller regions in which the localization is performed taking advantage of the probabilistic notion of *robust quadrilaterals*. A robust quad is a set of four nodes fully-connected by distance measurements and well-spaced in such a way that no ambiguity can arise, even when in the presence of noise. The algorithm, which does not require anchors, merges the sub-regions using a coordinate system registration procedure. Such a procedure maps local reference systems into a global one providing the best fitting matrix when in presence of a set of common nodes. An optional optimization step can be provided in order to refine the local map first. The weakness of this approach, as pointed out by the same authors, is that under conditions of low node connectivity or high measurement noise, the algorithm may be able to localize only a reduced number of nodes. In [43], the authors propose an approach where localization is performed by exploiting clustering information. Starting from locally-aware anchors, an initial set of calibrated nodes is built. This set is then expanded to include iteratively all the cluster-heads. Due to the iterative nature of this approach a refining step is required in order to provide reliable location estimates. Once the cluster-heads have been fully localized, the remaining follower nodes, i.e., non-cluster-head nodes, can be localized.

Range-free algorithms instead may offer a valid alternative anytime distance-information are not available, due perhaps to stringent hardware limitations. In [80], a range-free localization algorithm called APIT is proposed. In this work, the environment is first isolated into triangular regions defined by beacons and successively each node checks whether it is inside or outside of these regions to determine its location. Combinations of anchor positions can be used to reduce the diameter of the estimated area. Although an interesting insight on how localization error affects a variety of location-dependent applications such as geographical routing or target tracking is provided, an impractical number of beacons might be required to achieve satisfactory performances. In [158], a sequence-based RF localization algorithm called Ecolocation is proposed. The idea is to determine the location of unknown nodes by examining the ordered sequence of received signal strength (RSS) measurements taken at multiple reference nodes. The authors propose a constraint-based approach that provides for robust location decoding even in the presence of random RSS fluctuations due to multi-path fading and shadowing. However, the algorithm performance is heavily conditioned by the number of available reference nodes. In [34], the authors propose a RF-based distributed localization method where location are estimated by simply averaging the positions of all the anchors it is connected to. Obviously, the accuracy of the estimation is strictly related to the density of anchors deployed in the environment and the density required to obtain an acceptable estimation is fairly practical.

Anchor-free algorithms may finally represent a valid alternative solution in case prior knowledge about location are not available and an estimation in regards to a global reference frame is not required. In [126], the authors propose the Anchor-

Free Localization algorithm (AFL), an algorithm where all nodes concurrently calculate and refine their coordinate information. In detail, AFL is composed of two steps. During the first step, a folder-free graph embedding is computed starting from the original embedding and selecting five ad-hoc reference nodes used to approximate the polar coordinate of any other node. Successively, a mass-spring based optimization is performed in order to correct and balance localized errors. In [159], an anchor-free node localization protocol, which exploits clusterization to achieve scalability, is proposed. Such a protocol consists of three steps: network-bootstrapping, local position discovery and global localization. During the first step clusters are identified and a "breadth first spanning tree" rooted at the head of each cluster is performed. Since each node is able to measure distances from its neighbors (by exploiting some TOA technique) and a route exists from it to the cluster headset, all local distance information are sent to the cluster heads. This information will be used during the second step to build a local map at each cluster head. Finally, in the third step cluster heads collaborate in order to obtain a global map of the network. Such a global coordinate system can be built from the local maps by simply exploiting matrix rotations, translations and mirroring.

6.3 Part II Organization

The second part of the dissertation is focused on the Self-Localization problem in Sensor Networks. The remaining portion of Part II is organized as follows. In Chapter 7 the Interlaced Extended Kalman Filter formulation along with some experimental results is proposed. In Chapter 8 the distribued Interlaced Information Filter is described and experimental results are given. In Chapter 9 the two approaches are compared in terms of performances and computational complexity and conclusions are drawn. Note that, in order to make the treatment of each subject as self-contained as possible, each chapter is provided with a preliminary theoretical background, the description of the problem settings and the discussion of individual simulations and experimental results.

Chapter 7

Distributed Interlaced Extended Kalman Filter

In this chapter, following the collaborative processing paradigm typical of sensor network philosophy, a new distributed approach is proposed. The resulting Interlaced Kalman Filter consists of several reduced-order Kalman Filter implementations, each one run (on-board) by a node. At each time step, a node updates the estimate of its location by combining the latest observations with the latest estimates broadcasted by its neighbors. As a result, a flexible decentralized framework with an acceptable computational complexity is achieved. Moreover, it does not require any prior knowledge when an estimation on a relative coordinate system is desired and turns out to be very robust even in the presence of noisy distance measurements.

7.1 Interlaced Kalman Filter

The Interlaced Kalman Filter (IKF) has been proposed in [71] to reduce the computational load of the estimation process for a class of nonlinear systems. The fundamental idea of the IKF is derived from the multi-players dynamic game theory, where the optimal solution is given by letting each player choose its strategy as optimal response to the strategy chosen by the other players [118]. IKF is applied to nonlinear system that can be fully linearized by means of an appropriate partition of the state space variables. In this way IKF consists of p parallel KF implementations, each one devoted to estimating only a subset of the state variables, while considering the remaining parts as deterministic time varying parameters. The linearization error is partially alleviated increasing the noise covariance matrices [71].

For sake of clarity, let us consider a system whose model can be written as (for the first filter $i = 1$ and $j = 2$, while for the second $i = 2$ and $j = 1$)

$$\begin{array}{rcl} x_k^{(i)} & = & \tilde{A}_k^{(i)} x_{k-1}^{(i)} + f^{(i)}(x_{k-1}^{(j)}, u_k) + w(i)_k \\ z_k & = & C^{(i)}(x_k^{(j)}) x_k^{(i)} + d^{(i)}(x_k^{(j)}) + v_k^{(i)} \end{array} \tag{7.1}$$

where $\tilde{A}_k^{(i)} = A^{(i)} + F^{(ij)}(x_{k-1}^{(j)})$ and $w(i)_k \sim \mathcal{N}(0, Q_k)$, $v(i)_k \sim \mathcal{N}(0, R_k)$ are zero-mean uncorrelated white process noise vectors respectively characterized by the covariance matrices Q_k and R_k. The IKF equations proceed from KF filter

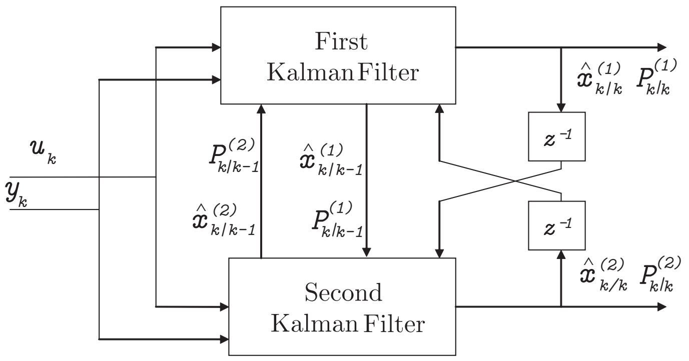

Figure 7.1: Interlaced Kalman Filter

equations, as shown in Fig. 7.1. At the k-th step, each sub-filter forms a prediction by exploiting its estimate along with the one provided by the other filters, according to the following equations:

$$\hat{x}^{(i)}_{k|k-1} = \tilde{A}^{(i)}_k \hat{x}^{(i)}_{k-1|k-1} + f^{(i)}(\hat{x}^{(j)}_{k-1|k-1}) \tag{7.2a}$$

$$P^{(i)}_{k|k-1} = \tilde{A}^{(i)}_k P^{(i)}_{k-1|k-1} \tilde{A}^{(i)^T}_k + \tilde{Q}^{(i)}_k \tag{7.2b}$$

where

$$\tilde{Q}^{(i)}_k = Q^{(i)}_k + [J^{F,(ij)}_{x,j} + J^{f,(i)}_{x,j}] P^{(j)}_{k-1|k-1} [J^{F,(ij)}_{x,j} + J^{f,(i)}_{x,j}]^T \tag{7.3}$$

being $J^{F,(ij)}_{x,j}$ and $J^{f,(i)}_{x,j}$ the Jacobians of the relations $F^{(ij)}(x^{(j)}_{k-1|k-1})x^{(i)}_{k-1|k-1}$ and $f^{(i)}(\cdot)$ with respect to $x^{(j)}_k$.

After the prediction step the estimates computed by the two sub-filters are exchanged and used during the update step. In this step the observation prediction is formed and compared with the measure z_k provided by the system

$$\hat{x}^{(i)}_{k|k} = \hat{x}^{(i)}_{k|k-1} + K^{(i)}_k [z_k - C^{(i)}(\hat{x}^{(j)}_{k|k-1})\hat{x}^{(i)}_{k|k-1} - d^{(i)}(\hat{x}^{(j)}_{k|k-1})] \tag{7.4a}$$

$$P^{(i)}_{k|k} = P^{(i)}_{k|k-1} - K^{(i)}_k C^{(i)}(\hat{x}^{(j)}_{k|k-1}) P^{(i)}_{k|k-1} \tag{7.4b}$$

where the Kalman gain is computed applying the relation

$$\begin{aligned} K^{(i)}_k &= P^{(i)}_{k|k-1} C^{(i)}(\hat{x}^{(j)}_{k|k-1})^T \cdot \\ &\cdot \; [C^{(i)}(\hat{x}^{(j)}_{k|k-1}) P^{(i)}_{k|k-1} C^{(i)}(\hat{x}^{(j)}_{k|k-1})^T + \tilde{R}^{(i)}_k]^{-1} \end{aligned} \tag{7.5}$$

in which

$$\tilde{R}^{(i)}_k = R_k + \left[J^{C,(i)}_{x,j} + J^{d,(i)}_{x,j}\right] P^{(j)}_{k|k-1} \left[J^{C,(i)}_{x,j} + J^{d,(i)}_{x,j}\right]^T \tag{7.6}$$

where $J^{C,(i)}_{x,j}$ and $J^{d,(i)}_{x,j}$ are the Jacobians of $C^{(i)}(x^{(j)}_{k|k-1})x^{(i)}_{k|k-1}$ and $d^{(i)}(\cdot)$ with respect to $x^{(j)}_k$.

From equations (7.3) and (7.6) can be noticed that the process and measurement noise covariance matrices $Q_k^{(i)}$ and $R_k^{(i)}$ are suitably increased by the addition of positive semi-definite quantities that take into account the error introduced by the decoupling operation. As shown in [133], indeed, it is easy to recognize that the term added to R_k in (7.6) represents the cross-correlation between the filters due to the innovation process, while the term added to $Q_k^{(i)}$ in (7.3) is related to the cross-correlation induced by the propagation process.

Note that, in a deterministic framework, sufficient conditions that guarantee the local convergence of the estimator are established in [72]. This formulation of IKF assumes that both substate transition mapping and observation mapping, i.e. equations (7.1), depend affinely on their arguments. If these assumptions are released, the algorithm can still be applied by applying linearization to every subsystem at each step. In this way, the Interlaced Extended Kalman Filter (IEKF) is obtained.

7.2 IEKF for Sensor Network

A distributed Interlaced Kalman Filter can be formulated in regard to the Sensor Network scenario described in Appendix C. Specifically, Ω parallel reduced-order EKFs are implemented. Each one runs on a node and is devoted to estimate its location. Due to the specific nature of the system, the filter is characterized by the prediction step of a linear KF and the correction step of an EKF.

7.2.1 Prediction

In the following a set of equations describing the prediction step of the proposed Interlaced Kalman Filter for the i-th node is given:

$$\begin{aligned} \hat{x}_{k|k-1}^{(i)} &= \hat{x}_{k-1|k-1}^{(i)} \\ P_{k|k-1}^{(i)} &= P_{k-1|k-1}^{(i)} + \tilde{Q}_k \\ \tilde{Q}_k^{(i)} &= Q_k^{(i)} \end{aligned} \tag{7.7}$$

Note that, the system is naturally fully decoupled, i.e. the state transition of each node does not affect the location of the other nodes. As a consequence, the term $\tilde{Q}_k^{(i)}$, described in (7.2b), does not have to be computed. The term $Q(k)$ is used instead.

7.2.2 Correction

In the following a set of equations describing the correction step of the proposed Interlaced Kalman Filter for the i-th node is given:

$$\begin{aligned}
\hat{x}^{(i)}_{k|k} &= x^{(i)}_{k|k-1} + K^{(i)}_k \nu^{(i)}_k \\
\nu^{(i)}_k &= z^{(i,j)}_k - h(\hat{x}^{(i)}_{k|k-1}, x^{(j)}_{k|k-1}) \quad \forall j \in \mathcal{N}(i) \\
K^{(i)}_k &= P^{(i)}_{k|k-1} {Jh^{(i)}_k}^T {S^{(i)}}^{-1}_k \\
S^{(i)}_k &= Jh^{(i)}_k P^{(i)}_{k|k-1} {Jh^{(i)}_k}^T + \tilde{R}^{(i)}_k \\
P^{(i)}_{k|k} &= (I - K_k Jh^{(i)}_k) P^{(i)}_{k|k-1} \\
\tilde{R}^{(i)}_k &= diag\{R^{(i)}_k + Jh^{(j)}_k P^{(j)}_{k|k-1} {Jh^{(j)}_k}^T\} \quad \forall j \in \mathcal{N}(i)
\end{aligned} \tag{7.8}$$

where $\mathcal{N}(i)$ is the neighborhood of the i-th node. Note that, due to the non-linearity of the mapping, the Jacobian of the maps $h(\cdot)$ described in C.6 have to be used, instead of matrix $C^{(i)}(\cdot)$, in equations (7.4) and (7.6). Moreover, when a node detects another one, the covariance update is calculated according to eq. (7.4b). However, since the position of an anchor does not affect the location of the node, the term $\tilde{R}^{(i)}_k$ is not computed in (7.6). The term $R(k)$ is used instead.

7.2.3 Complexity analysis

Providing an analysis of the computational complexity is particularly important in all contexts where the limited hardware capabilities call for effective algorithm implementations. Indeed, it is well known that the main drawbacks related with the implementation of localization algorithms based on EKF approaches are due to huge computational load and memory occupancy.

Let us assume N to be the dimension of the state space and M the number of observations available (neighbors of a node), with $N < M$. The dominant operation at each iteration is the inversion of the innovation (or residual) covariance matrix S. Now, since $S \in \mathcal{R}^{M \times M}$ the complexity results $\mathcal{O}(M^3)$ when a naive implementation of the matrix inversion is considered. However, it may be reduced to $\mathcal{O}(M^{2.376})$ according to the analysis provided in [51].

If a slight decrease of the estimation accuracy is acceptable, this computational complexity can be even further reduced. This can be achieved by updating the estimate considering a single observation at a time. This way, the complexity of the filter becomes $\mathcal{O}(NM)$, where $\mathcal{O}(N)$ is the complexity for each update and M is the number of time it is required.

Finally, since the dimension of the state space is fixed ($N = 2$), the complexity scales as $\mathcal{O}(M)$, which is indeed an interesting trade-off between estimation accuracy and computational requirements suitable with the limited hardware resources of nodes.

7.3 Performance Evaluation

The proposed distributed Interlaced Kalman Filter has been thoroughly investigated in both simulated environments and with real data. Experiments carried out with MICAz (MPR2400) platform showed the capability of the algorithm in several real contexts, while simulations were fundamental to investigate its scalability. The exposition is organized in two parts: the first analyzes the behavior of the framework within a simulate context while the second reports some results with real data.

7.3.1 Problem Setting

Computer Simulations Simulations have been carried out in a framework developed under Matlab by the authors. This framework provides an implementation of the sensor network scenario modeling given in Appendix C. It supports both a complete simulated context as well a test-bed to run data coming from real hardware. These two different operative modalities turn out to be very useful, both to test the correctness and the effectiveness of the algorithm.

Real Context The network has been built with the MICAz (MPR2400) platform, a generation of Motes from Crossbow Technology. The MPR2400 (2400 MHz to 2483.5 MHz band) uses the Chipcon CC2420, IEEE 802.15.4 compliant, ZigBee ready radio frequency transceiver integrated with an Atmega128L micro-controller. It provides also a flash serial memory, as well as a 51 pin I/O connector that allows several sensor and data acquiring boards to be connected to it. MICAz platform comes along with TinyOS, an open-source event-driven operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS component library includes network protocols, distributed services, sensor drivers, and data acquisition tools, all of which can be used as–is or be further refined for a custom application.

7.3.2 Evaluation Criteria

In order to evaluate the effectiveness of the proposed algorithm, two indexes of quality have been taken into account:

- Estimation Accuracy.
- Convergence Velocity.

The estimation accuracy is expressed in terms of distance between the real node location and the estimated one provided by the filter. The Euclidian distance is exploited as metric. In particular, maximum, minimum and average error are given so that a comprehensive view of the situation is obtained. The velocity

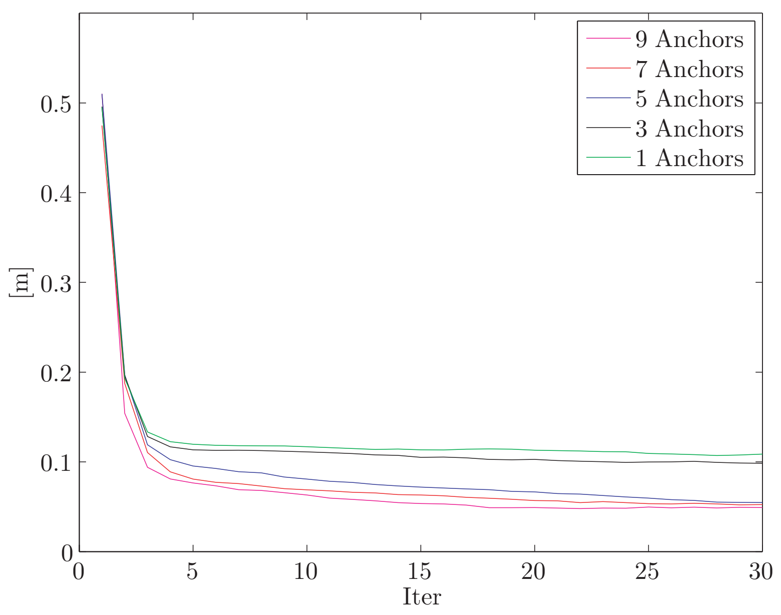

Figure 7.2: Estimation error vs. Density of anchor deployment: variable number of anchors (from 1 to 9), fixed number of nodes (70).

of convergence is given in terms of the number of iterations required for the algorithm in order to stabilize around a certain value.

7.3.3 Simulation

Simulations have been carried out in order to investigate the scalability of the proposed IEKF. Particular attention has been devoted to the robustness and accuracy of the estimation. In detail, the following aspects have been taken into account:

- Density of anchor deployment.
- Density of node deployment.
- Level of noise of observations.

Fig. 7.2 shows the result when considering a variable number of anchors, ranging from 1 to 9, with a fixed number of nodes 70. According to this result, the algorithm performs better in terms of estimation accuracy and convergence rate when considering an increasing number of anchors. In detail two different behaviors can be recognized, considering anchors ranging from 1 to 3 or from 5 to 9. This provides a way to define the optimal number of anchors to be used for a real deployment with respect to some parameters of interest.

Fig. 7.3 shows the result when considering a variable number of nodes, ranging from 10 to 90, with a fixed number of anchors 5. According to this result, the algorithm performs slightly better, in terms of convergence rate, when considering

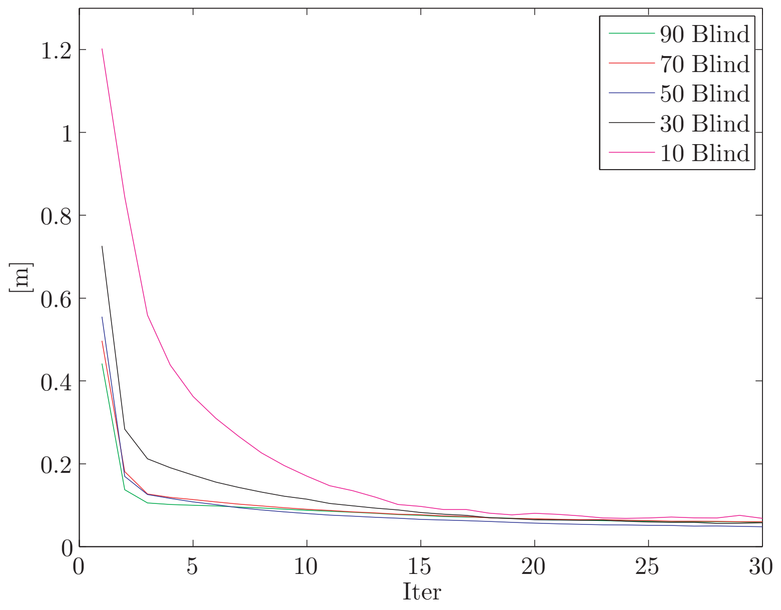

Figure 7.3: Estimation error vs. Density of nodes deployment: variable number of nodes (from 10 to 90), fixed number of anchors (5).

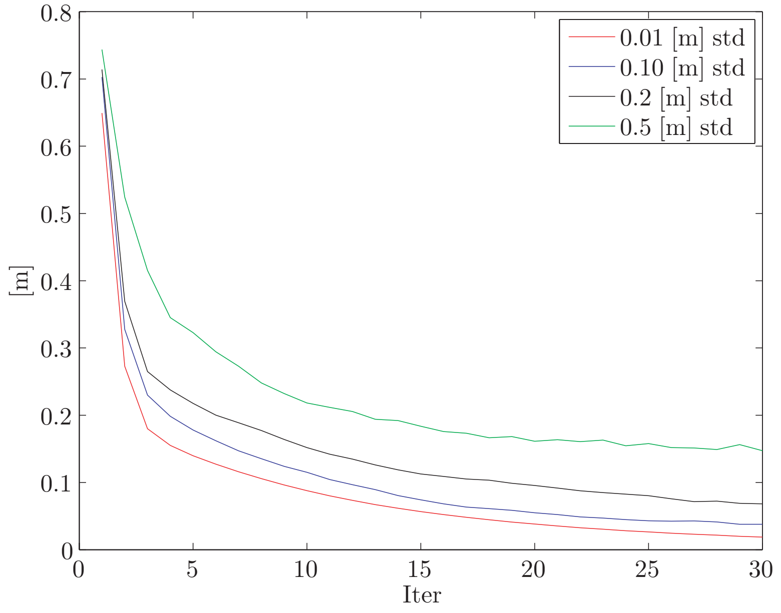

Figure 7.4: Estimation error vs. Level of noise of observations: variable level of noise (std ranging from 0.01m to 0.5m), fixed number of anchors (5), fixed number of anchors (30)

an increasing number of nodes. However, the accuracy of the estimations seems not to be influenced by the number of nodes available. This can be explained by the fact that the accuracy is mainly related to the number of available anchors and the noise of observations.

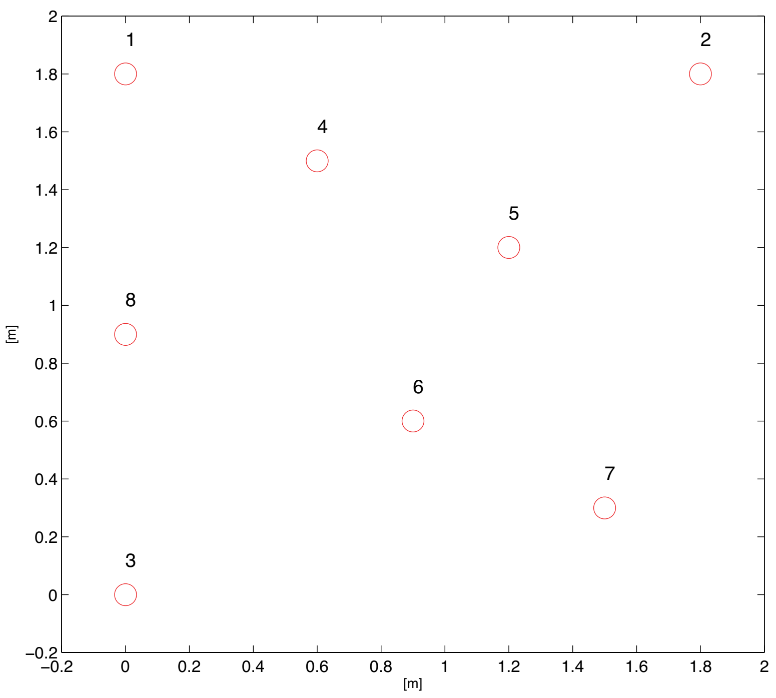

Figure 7.5: First configuration for the network deployment: anchors (1-3) were arranged on the border, while nodes to be localized (4-8) were randomly positioned.

Fig. 7.4 shows the result when considering a variable level of noise with a standard deviation ranging from 0.01 m to 0.5 m, with both a fixed number of anchors 5 and nodes 30. According to this result, the algorithm performs better, in terms of convergence rate, when considering a decreasing level of noise.

7.3.4 Experimental Results

Real experiments involved several indoor environments of the robotics laboratory of University of "Roma Tre" for the network deployment. Two different configurations have been built and several data acquisitions have been done. Moreover, different anchors arrangements have been considered to better understand whether the performances might be influenced by the anchor locations. Real locations were measured manually taking advantage of the regularity of the flooring grid.

Specifically, Fig. 7.5 describes the first configuration that has been considered. Eight nodes have been deployed and three different sets of three anchors have been considered, namely $\{1, 2, 3\}, \{2, 7, 8\}, \{3, 4, 5\}$. In addition, each node was ideally within the communication range of each other so that a full connected graph was available.

Further, Fig. 7.6 describes the second configuration that has been exploited. In this case, ten nodes were deployed and three different sets of anchors have been considered, i.e., $\{1, 2, 3\}, \{1, 6, 7\}, \{5, 9, 10\}$. Again, each node was ideally

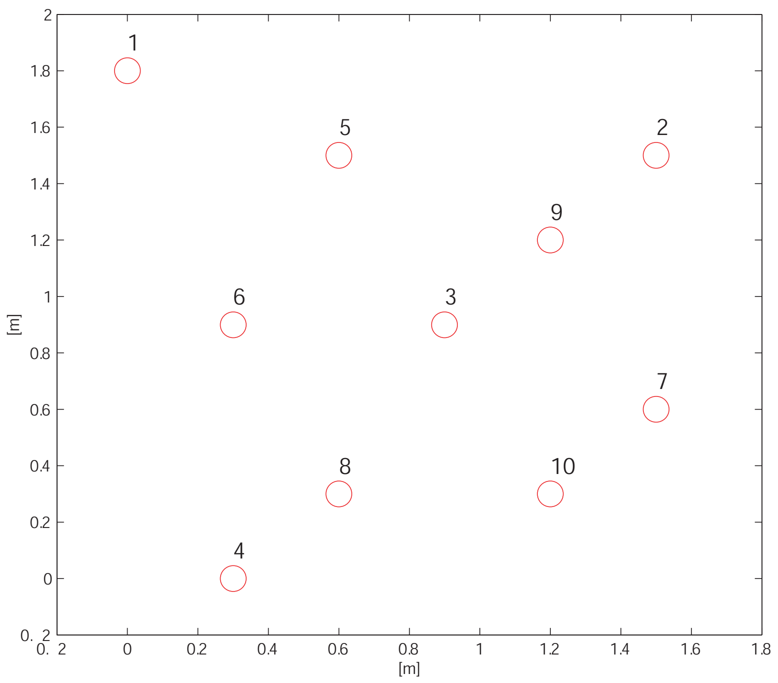

Figure 7.6: Second configuration for the network deployment: anchors (1-3) as well as nodes to be localized (4-10) were almost randomly arranged.

within the range of communication of each other in order to have a full connected graph.

A comparison against a centralized version of an Extended Kalman Filter is provided. It aims to understand the advantages as well as the drawbacks that arise when decentralizing an algorithm. In particular, 100 trials were run for each configuration. Afterwards, the collected data was used to compute the indexes of interest given in 7.3.2.

IEKF vs. EKF

Here a comparison of the proposed Interlaced Extended Kalman Filter (IEKF) against a centralized Extended Kalman Filter (EKF) is provided. In regards to the first configuration shown in Fig. 7.5, the results of the centralized and distributed algorithms averaged over 100 trials are collected in Table 7.1. Here a synoptic comparison between the two approaches can be found. According to the accuracy of available data, both algorithms are able to localize all nodes within the network with similar performances. As expected, EKF performs slightly better, especially in terms of maximum error and convergence velocity. This is related to the more comprehensive interpretation of data typical of a centralized approach which always takes advantage of the complete knowledge of cross-correlation terms. These terms at the same time influence the convergence rate which is quicker compared to the IEKF.

Extended Kalman Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	7.85	2.84	5.95	12
{2,7,8}	11.28	3.24	7.33	10
{3,4,5}	8.81	6.31	7.99	12

Distributed Interlaced Extended Kalman Filterr				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	8.12	3.18	6.36	15
{2,7,8}	11.49	3.92	6.14	13
{3,4,5}	10.17	3.87	7.48	29

Table 7.1: First Deployment. Centralized EKF vs. Distributed IEKF.

Note that, for both algorithms better results were achieved when anchors were deployed along the borders. This is particularly crucial in a distributed context where the lack of data, locally experienced by each node, needs to be balanced by its consistency. Indeed, this arrangement reduces the possibility of symmetrical solutions which might fit data properly.

Extended Kalman Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	10.65	4.88	7.6	18
{1,6,7}	10.47	2.62	5.73	13
{5,9,10}	9.76	1.06	6.88	17

Distributed Interlaced Extended Kalman Filterr				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	11.18	4.09	7.96	25
{1,6,7}	11.06	2.57	6.59	26
{5,9,10}	12.71	1.96	5.63	24

Table 7.2: Second Deployment. Centralized EKF vs. Distributed IEKF.

In regards to the second configuration shown in Fig. 7.6, the results of the centralized and distributed algorithms averaged over 100 trials are collected in Table 7.2. Also in this case the EKF performs slightly better compared to the IEKF. However the IEKF, apart from all the advantages deriving from it distributed nature, still provides effective estimations with a reasonable accuracy.

As far as the computational complexity is concerned, Table 7.3 shows the performance for both the distributed IEKF and the centralized EKF. Due to the small instance of the localization problem, the difference of the computational load cannot be appreciated. It should be noticed, however, that the IEKF update can be split in several smaller updates running independently on different processors (one for each node that needs to be localized), while the same parallelism cannot be achieved by EKF.

Table 7.3: Full update time over an Intel Pentium M 725 (1.6GHz)

Conf.	EKF [s]	IEKF [s]
1	0.0024	0.0021
2	0.0038	0.0027

7.4 Considerations

In this chapter a new approach for the self-localization problem is proposed. A distributed Interlaced Extended Kalman Filter has been developed by following the collaborative processing paradigm typical of sensor network philosophy. It consists of a set of reduced-order Kalman Filters, each one run by a node which collaborates in order to provide a reliable estimation for the whole network. The fundamental idea of the IKF is derived from the multi-players dynamic game theory, where the optimal solution is given by letting each player choose its strategy as optimal response to the strategy chosen by the other players. In particular, each node iteratively updates its estimate by combining its latest observations (in terms of inter-node distances) with the latest estimates broadcasted by its neighbors. Several simulations were performed to investigate the scalability of the approach in regard to the density of anchors, the density of the nodes and with respect to the level of noise of observations. Moreover, real experiments involving MICAz platform were performed to validate the capability of the proposed approach in a real context. In addition, a comparison of the proposed IEKF against a centralized formulation of the EKF was performed to investigate both advantages and drawbacks concerning a distributed implementation. According to the experimental results, the centralized EKF performs in average slightly better. This can be explained by the different amount of data available at each iteration for the centralized formulation compared to the distributed formulation. Indeed, this is particularly crucial for any adverse contexts, e.g., a collinear anchor deployment, where such a lack of information results in a severe deterioration of the performances.

Chapter 8

Distributed Extended Information Filter

In this chapter a new approach based on Information Theory for the self-localization problem is proposed. A Decentralized Extended Information filter is built starting from a centralized formulation by means of some simplifying assumptions. Each node runs a reduced-order filter to estimate its location. This is achieved locally by combining the most recent observation with the latest estimates provided to each node by its neighbors. Note that, an Information Filter is essentially a Kalman Filter expressed in terms of measures of information about the parameters (state) of interest (rather than direct state estimates and their associated covariance). However, moving to this dual domain may result in an effective reduction of the computational complexity when some specific assumptions are met.

8.1 The Information Filter

An Information Filter (IF) is essentially a Kalman Filter (KF) expressed in terms of measures of *information* about the parameters (state) of interest rather than direct state estimates and their associated covariances [75]. The two key information-analytic variables are the *information matrix* and the *information state vector*, where the term information is used according to the Fisher definition.

8.1.1 Theoretical Formulation

The Fisher information matrix Y_k is the amount of information that an observable random variable z carries about an unobservable parameter x upon which the likelihood function of z, $L(x) = p(z \mid x)$, depends. It can be derived as the covariance of the *score function*, that is the partial derivative, with respect to some parameter x, of the logarithm (commonly the natural logarithm) of the likelihood function. If the observation is z and its likelihood is $L(x) = p(z \mid x)$, then the score $\mathcal{S}_k(x)$ can be described as follows:

$$\begin{aligned}\mathcal{S}_k(x) &= \nabla_x \ln p(z_k \mid x_k) &(8.1)\\ &= \frac{\nabla_x\, p(z_k \mid x_k)}{p(z_k \mid x_k)} &(8.2)\end{aligned}$$

Moreover, being the expectation of the score:

$$\begin{aligned}E[\mathcal{S}_k(x)] &= \int \frac{\nabla_x\, p(z_k \mid x_k)}{p(z_k \mid x_k)}\, p(z_k \mid x_k)\, dz_k &(8.3)\\ &= \nabla_x \int p(z_k \mid x_k)\, dz_k &(8.4)\\ &= \nabla_x\, 1 = 0 &(8.5)\end{aligned}$$

the Information matrix Y_k is simply the second order moment of the score function $\mathcal{S}_k(x)$, as follows:

$$\begin{aligned}Y_k &= E[\mathcal{S}_k(x)\mathcal{S}_k(x)^T] &(8.6)\\ &= E[\{\nabla_x \ln p(z_k \mid x_k)\}\{\nabla_x \ln p(z_k \mid x_k)\}^T] &(8.7)\end{aligned}$$

Furthermore, if the following regularity condition holds:

$$\int H_x(p(z_k \mid x_k)) = \nabla_x \nabla_x^T p(z_k \mid x_k) = 0 \quad (8.8)$$

where H_x is the square matrix of second-order partial derivatives (i.e., Hessian Matrix), the Information matrix Y_k can be also written as:

$$Y_k = -E[\nabla_x \nabla_x^T \ln p(z_k \mid x_k)] \quad (8.9)$$

At this point, when the likelihood function $p(z \mid x)$ is a Gaussian distribution and the posterior conditional distribution is Gaussian as well, described as $p(x_k \mid z) \sim \mathcal{N}(\hat{x}_k, P_k)$, then it can be proved [117] that the Information Matrix is equal to the inverse of the covariance matrix P_k as follows:

$$Y_k = P_k^{-1}. \quad (8.10)$$

Likewise, the information state vector y_k can be easily derived as the product of the inverse of the information matrix and the state estimate as follows:

$$\begin{aligned}y_k &= Y_k\, \hat{x}_k &(8.11)\\ &= P_k^{-1}\, \hat{x}_k. &(8.12)\end{aligned}$$

8.1.2 The Information Filter: Algorithmic Derivation

The Information Filter formulation can be easily derived from the Kalman Filter formulation under the assumption of Gaussianity previously stated. The Kalman filter is essentially a set of mathematical equations that implements a predictor-corrector type estimator. It can be proven to be optimal in the sense that it minimizes the estimated error covariance when some presumed conditions are met [107]. The two key-variables are the estimate of the state of interest and its associated covariance, respectively $\hat{x}$ and P.

Given a dynamical system whose model is described the following equations:

$$\begin{aligned} x_k &= F_k\, x_{k-1} + B_k\, u_{k-1} + w_{k-1} & (8.13)\\ z_k &= H_k\, x_k + v_k & (8.14) \end{aligned}$$

where x_k is the state of the system of interest at time k, F_k is the state transition matrix from time $k-1$ to time k, B_k is the input matrix, u_{k-1} is the input control vector, $w_k \sim \mathcal{N}(0, Q_k)$ is the related Gaussian with zero mean and uncorrelated process noise, H_k is the observation matrix and $v_k \sim \mathcal{N}(0, R_k)$ is the related Gaussian with zero mean and uncorrelated observation noise. The Kalman Filter implementation can be summarized by the following set of equations:

- Prediction

$$\begin{aligned} \hat{x}_{k|k-1} &= F_k \hat{x}_{k-1|k-1} \\ P_{k|k-1} &= [F_k P_{k-1|k-1} F_k^T + Q_k] \end{aligned} \tag{8.15}$$

- Correction

$$\begin{aligned} \hat{x}_{k|k} &= x_{k|k-1} + K_k \nu_k \\ \nu_k &= z_k - H_k \hat{x}_{k|k-1} \\ K_k &= P_{k|k-1} H_k^T S_k^{-1} \\ S_k &= H_k P_{k|k-1} H_k^T + R_k \\ P_{k|k} &= (I - K_k H_k) P_{k|k-1} \end{aligned} \tag{8.16}$$

where $P_{k|k}$ and $\hat{x}_{k|k}$ are respectively the covariance matrix and the state of interest at time i given information up to time j, according to the Barshalon notation [14]. The Information Filter implementation can be obtained from the previous set of equations by performing the substitutions given in eq. (8.12) and eq. (8.10) as follows:

- Prediction

$$\begin{aligned} Y_{k|k-1} &= [F_k Y_{k-1|k-1}^{-1} F_k^T + Q_k]^{-1} \\ L_{k|k-1} &= Y_{k|k-1} F_k Y_{k-1|k-1}^{-1} \\ y_{k|k-1} &= L_{k|k-1} y_{k-1|k-1} \end{aligned} \tag{8.17}$$

- Estimation

$$\begin{aligned} Y_{k|k} &= Y_{k|k-1} + I_k \\ y_{k|k} &= y_{k|k-1} + i_k \\ I_k &= H_k^T R_k^{-1} H_k \\ i_k &= H_k^T R^{-1} z_k \end{aligned} \tag{8.18}$$

.

A more comprehensive description of the Information Filter derivation is given in [117].

8.2 The Extended Information Filter for Sensor Networks

Due to the nonlinear nature of the observation model, the linear Information Filter previously introduced cannot be applied as it is. An extension to deal with the non-linearity of the observation model is required. Note that having a linear prediction model results in a "hybrid" Information Filter: with the prediction equation of a linear IF and the estimation equation of an Extended IF. In the following, a centralized formulation of the filter is proposed. Then, a decentralized one based on simplifying assumptions is devised. However, both filters can be summarized by the same two-stage formulation:

- Prediction

$$\begin{aligned} Y_{k|k-1} &= [Y_{k-1|k-1}^{-1} + Q_k]^{-1} \\ L_{k|k-1} &= Y_{k|k-1} Y_{k-1|k-1}^{-1} \\ y_{k|k-1} &= L_{k|k-1} y_{k-1|k-1} \end{aligned} \tag{8.19}$$

- Estimation

$$\begin{aligned} Y_{k|k} &= Y_{k|k-1} + I_k \\ y_{k|k} &= y_{k|k-1} + i_k \\ I_k &= JH_k^T R_k^{-1} JH_k \\ i_k &= JH_k^T R^{-1} z'_k \\ z'_k &= \nu_k + JH_k \hat{x}_{k|k-1} \\ \nu_k &= z_k - h(\hat{x}_{k|k-1}) \end{aligned} \tag{8.20}$$

Differences between the centralized formulation and the distributed formulation are merely related to the state space dimension and to the construction of the Jacobian matrix JH.

8.2.1 Centralized Formulation

In order to derive the extended version of the estimation step, the Jacobian matrix JH with respect to the observation model has to be computed. Given a generic observation $z_k^{(i,j)}$, representing the distance from the node i to the node j measured by the node i, the related Jacobian row is:

$$Jh^{(i,j)} = [\cdots \underbrace{\frac{\partial h(i,j)}{\partial x_i} \frac{\partial h(i,j)}{\partial y_i}}_{i} \cdots \underbrace{\frac{\partial h(i,j)}{\partial x_j} \frac{\partial h(i,j)}{\partial y_j}}_{j} \cdots] \tag{8.21}$$

where:

$$\begin{aligned} \frac{\partial h(i,j)}{\partial x_i} &= \frac{x_i}{\sqrt{(x_i+x_j)^2+(y_i+y_j)^2}} \\ \frac{\partial h(i,j)}{\partial y_i} &= \frac{y_i}{\sqrt{(x_i+x_j)^2+(y_i+y_j)^2}} \end{aligned}$$

Now, given a subset of M nodes, all of them within the range of visibility of each other, the Jacobian matrix $Jh(i)$ for a generic node i is given by:

$$\begin{aligned} Jh^{(i)} &= \begin{bmatrix} Jh^{(i,1)} \\ \vdots \\ Jh^{(i,k)} \\ \vdots \\ Jh^{(i,M)}] \end{bmatrix} \qquad (8.22) \\ &= \begin{bmatrix} \frac{\partial h(i,1)}{\partial x_1} \frac{\partial h(i,1)}{\partial y_1} & \cdots & \frac{\partial h(i,1)}{\partial x_i} \frac{\partial h(i,1)}{\partial y_i} & \cdots & \cdots \\ \vdots & & & & \vdots \\ \cdots & \frac{\partial h(i,k)}{\partial x_k} \frac{\partial h(i,k)}{\partial y_k} & \frac{\partial h(i,k)}{\partial x_i} \frac{\partial h(i,k)}{\partial y_i} & \cdots & \cdots \\ \vdots & & & & \vdots \\ \cdots & \cdots & \frac{\partial h(i,M)}{\partial x_i} \frac{\partial h(i,M)}{\partial y_i} & \cdots & \frac{\partial h(i,M)}{\partial x_M} \frac{\partial h(i,M)}{\partial y_M} \end{bmatrix} \end{aligned}$$

Therefore, the complete Jacobian matrix JH is given by the juxtaposition of the Jacobian matrix computed for each node as follows:

$$JH = [Jh^{(1)} \cdots Jh^{(i)} \cdots Jh^{(\Omega)}]^T \tag{8.23}$$

where the temporal index k has been omitted while describing the Jacobian matrix for the sake of clarity.

8.2.2 Decentralized Formulation

A decentralized formulation can be derived from the centralized formulation previously stated by means of some simplifying assumptions. As previously pointed out, the system model is linear and fully decoupled thus suitable for a decentralized implementation. Conversely, the Jacobian Matrix $Jh^{(i)}$, defined for each node i in eq. 8.24, features some couplings that require approximations to decentralize the computation. Specifically, for each node i the other nodes of the network are assumed to be landmarks. As a consequence, for a generic Jacobian row $Jh^{(i,j)}$, the partial derivatives of node j are always naughts. Therefore, the related Jacobian Matrix $Jh^{(i)}$ for a given node i is:

$$Jh^{(i)} = \begin{bmatrix} Jh^{(i,1)} \\ \vdots \\ Jh^{(i,k)} \\ \vdots \\ Jh^{(i,M)}] \end{bmatrix} = \begin{bmatrix} \cdots & \frac{\partial h(i,1)}{\partial x_i} \; \frac{\partial h(i,1)}{\partial y_i} & \cdots \\ & \vdots & \\ \cdots & \frac{\partial h(i,k)}{\partial x_i} \; \frac{\partial h(i,k)}{\partial y_i} & \cdots \\ & \vdots & \\ \cdots & \frac{\partial h(i,M)}{\partial x_i} \; \frac{\partial h(i,M)}{\partial y_i} & \cdots \end{bmatrix} \tag{8.24}$$

In this way the complete Jacobian matrix JH, described in eq. 8.23, turns out to be a block-matrix. Therefore, it can be easily decomposed in order to have several reduced-order filters. In particular, each node runs a reduced-order filter with the aim of estimating its location with respect to information in terms of observations and latest estimates coming from the other nodes.

8.2.3 Complexity Analysis

The analysis of the computational complexity is important to validate the performance of an algorithm. In this case, due to the limited storage, processing and communication capabilities typical of this hardware, it becomes even more crucial.

Let N be the dimension of the state space of a node and M the number of its neighbors, with $M >> N$. For a given node, at each time step k the dominant operation is the computation of the information matrix $I(k)$. It consists of the inversion of a (diagonal) matrix and two multiplications between matrixes. Since the Jacobian $JH \in \mathcal{R}^{M \times N}$, the overall computational complexity is $\mathcal{O}(N^2 M)$. Note that, this complexity is significantly lowered by the diagonal nature of the covariance matrix R. In fact, both inversion and multiplication involving a diagonal matrix have a complexity which scales with the size of the matrix.

Finally, since the dimension of the state is fixed ($N = 2$), the computational complexity scales as $\mathcal{O}(M)$. Therefore, it is suitable for an online implementation even when in presence of hardware with limited capabilities.

8.3 Performance Evaluation

The proposed distributed Extended Information Filter has been thoroughly investigated in both simulated environments and with real data. Experiments carried out with MICAz (MPR2400) platform showed the capability of the algorithm in several real contexts, while simulations were fundamental to investigate the scalability of the framework. The exposition is organized in two parts: the first analyzes the behavior of the framework within a simulated context while the second reports some results with real data.

8.3.1 Problem Setting

Computer Simulations Simulations have been carried out in a framework developed under Matlab by the authors. This framework provides an implementation of the sensor network scenario modeling given in Appendix C. It supports both a complete simulated context as well a test-bed to run data coming from real hardware. These two different operative modalities turn out to be very useful, both to test the correctness and the effectiveness of the algorithm.

Real Context The network has been built with the MICAz (MPR2400) platform, a generation of Motes from Crossbow Technology. The MPR2400 (2400 MHz to 2483.5 MHz band) uses the Chipcon CC2420, IEEE 802.15.4 compliant, ZigBee ready radio frequency transceiver integrated with an Atmega128L micro-controller. It provides also a flash serial memory, as well as a 51 pin I/O connector that allows several sensor and data acquiring boards to be connected to it. MICAz platform comes along with TinyOS, an open-source event-driven operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS component library includes network protocols, distributed services, sensor drivers, and data acquisition tools, all of which can be used as–is or be further refined for a custom application.

8.3.2 Evaluation Criteria

In order to investigate the effectiveness of the proposed localization technique, two indexes of quality are exploited considering several anchors deployment:

- Estimation Accuracy.
- Convergence Velocity.

The accuracy of the estimation is given in terms of distance between the estimated and the real location of a node. The Euclidian distance is adopted as metric. Maximum, minimum and average errors computed over the whole network are considered. The velocity of convergence is given in terms of the number of steps required by the algorithm to stabilize around the best estimation. This index provides an evaluation of the "reactivity" of the algorithm.

8.3.3 Simulation Results

Simulations have been performed to investigate the scalability of the proposed distributed Information Filter with respect to some factors of interest. A network of 50 nodes, randomly deployed in an 10×10m environment was considered.

Two factors were made varying and their influence on both the accuracy of the estimation and the velocity of convergence was evaluated:

- Observation Noise.
- Anchor Density.

The default setting for these factors were $\mathcal{N}(0, \sigma)$ with $\sigma = 15$ cm for the observation noise and 10 the number of anchors randomly deployed.

$\mathcal{N}(0,\sigma)$ [cm]	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Convergence [steps]
5	2.98	0.21	1.19	35
10	5.05	0.27	2.33	37
20	9.46	0.70	4.06	40

Table 8.1: Scalabiliy Analysis w.r.t. Observation Noise.

Number of Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Convergence [steps]
10	5.52	0.36	2.19	33
15	5.35	0.30	2.16	31
20	5.02	0.41	2.15	20

Table 8.2: Scalabiliy Analysis w.r.t. Anchors Density.

Tab. 8.1 shows the simulation results when changing the variance of the observation noise, while keeping the number of anchors fixed to 10. In particular, the accuracy of the algorithm decreases when increasing the variance of the observation noise. Likewise, the number of steps required for the convergence of the algorithm slightly increases as well.

Tab. 8.2 shows the simulation results when varying the number of anchors while keeping the variance of the observation noise fixed to $\sigma = 15$ cm. Note that, the higher the number of anchors, the quicker the convergence of the algorithm. In addition, a minor improvement of the estimation accuracy is experienced when increasing the number of anchors.

8.3.4 Experimental Results

Real experiments involving two different sensor network deployments are reported. The network was arranged taking advantage of the regularity of the flooring grid and real locations were manually measured exploiting such a regularity. Different anchors configurations have been investigated for each configuration.

Fig. 8.1 shows the first deployment where 10 nodes were considered. Each node was ideally within the communication range of each other. This way a full connected graph was achieved.

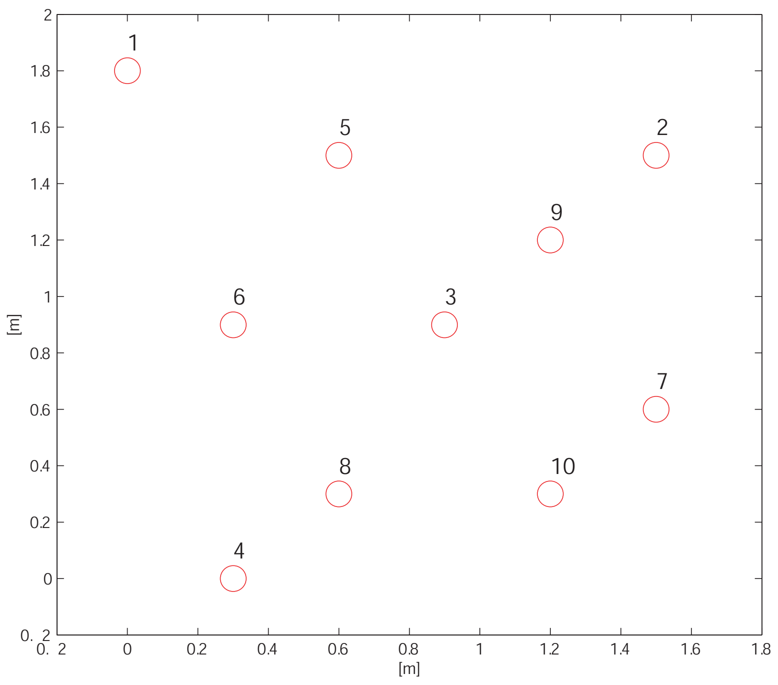

Figure 8.1: First configuration for the network deployement.

Fig. 8.2 shows the second deployment where 11 nodes were considered. Again, each node was ideally within the communication range of each other in order to have a full connected graph.

A comparison between the distributed formulation and the centralized formulation is provided. It aims to evaluate how the assumption required for distributing the algorithm may affect its performances.

Decentralized EIF vs Centralized EIF

Table 8.3 describes the result of the experiment involving the first environment (Fig. 8.1). Three different arrangements of anchors were considered, each one involving three nodes. According to experimental results for this configuration, varying the set of anchors does not significantly influence the accuracy of estimation. This holds for both the centralized and distributed formulation. In regard to the convergence velocity, the centralized formulation turns out to perform better. This can be explained by the fact that within the distributed formulation each node has only a local view of the global situation. Therefore, each estimate is computed by only partially exploiting the whole amount of information available.

Table 8.4 describes the result of the experiment involving the second environment (Fig. 8.2). Three different arrangements of anchors were considered again, each one involving three nodes. A deterioration of the performance is experienced for the decentralized formulation with respect to the third one. Indeed, the cen-

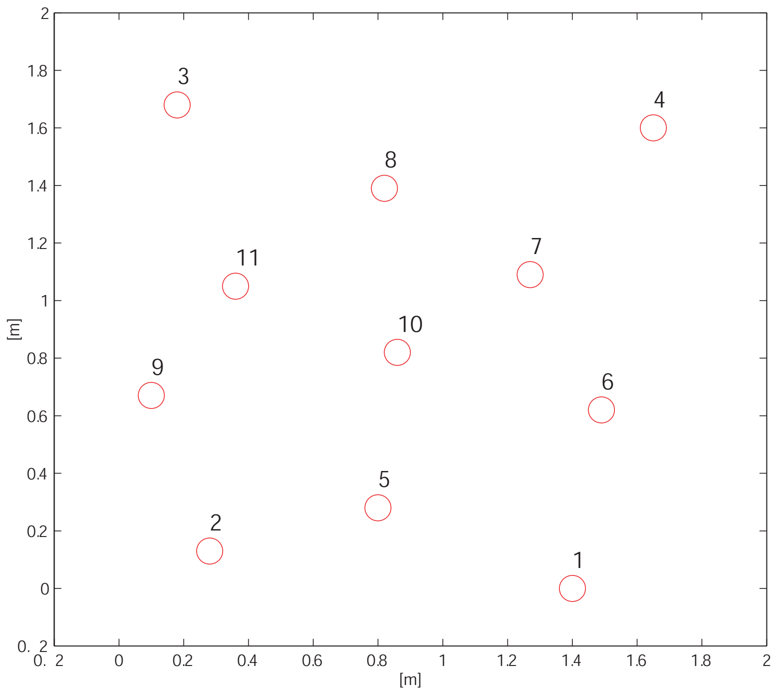

Figure 8.2: Second configuration for the network deployment.

Centralized Extended Information Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	12.09	2.86	6.41	5
{1,6,7}	10.93	1.90	5.88	12
{5,9,10}	13.13	3.37	7.93	13

Distributed Extended Information Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	13.13	4.28	8.70	24
{1,6,7}	13.56	1.86	7.21	33
{5,9,10}	13.44	1.93	5.64	21

Table 8.3: First Deployment. Centralized EIF vs Decentralized EIF.

Centralized Extended Information Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,4}	23.93	2.71	11.68	7
{3,4,5}	25.08	4.31	11.22	6
{6,10,11}	18.47	5.84	10.95	10

Distributed Extended Information Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,4}	18.12	3.97	13.75	34
{3,4,5}	22.26	10.86	15.12	33
{6,10,11}	31.7	7.62	17.14	35

Table 8.4: Second Deployment. Centralized EIF vs Decentralized EIF.

tralized formulation is able to overcome the weakness of such a collinear anchors deployment. It is attained by properly exploiting the amount of information available. However, this cannot be achieved by the decentralized formulation.

Note that, the algorithm may converge to an alternative admissible solution. Indeed, given Ω nodes with Θ anchors, “symmetrical” solutions may exist with respect to the deployment of the anchors. In the case of perfect communication among nodes, i.e., no packet is lost, placing anchors on the boundary is a sufficient condition to have an unique solution when full connectivity among nodes is available. However, in a real system some nodes may not be able to communicate with other nodes. Therefore in practice, alternative plausible solutions in regard to the available data may always exist. Experiments pointed out that some of the anchors deployment are more critical than others in terms of convergence capability.

8.4 Considerations

In this chapter a new approach based on Information Theory for the self-localization is proposed. Starting from a centralized formulation, a distributed Extended Information Filter (EIF) is built by means of some simplifying approximations. Each node runs a reduced-order EIF which aims to estimate its location. At each iteration, a node updates its estimate by combining its latest observations with the most recent estimates broadcasted by its neighbors. Simulations along with real experiments have been provided to investigate the effectiveness of the approach. Simulations showed the scalability of the algorithm with respect to the observation noise and the density of available anchors. While real experiments, carried out exploiting the MICAz (MPR2400) platform (a generation of Motes from Crossbow Technology), validated the algorithm capability in a real context. In order to investigate the robustness of the proposed approach, several anchors arrangement were considered. Experimental results showed that unless a particular situation is met, e.g. collinear anchor deployment, varying the set of anchors does not significantly influence the accuracy of estimation. Finally, a comparison of the distributed formulation against the centralized formulation is provided in order to understand how the simplifying assumptions might influence the performance. Experimental results showed that the centralized formulation is able to better overcome issues related to the anchors deployment. In addition, a quicker convergence is experienced for the centralized formulation. However, the distributed formulation still provides satisfactory performance along with all the advantages derived from distributing the computation.

Chapter 9

Algorithms Discussion and Comparison

In the previous chapters 7 and 8 two approaches for the Self-Localization problem in Sensor Networks have been proposed. The first, inspired by the multi-players dynamic game theory, consists of a set of reduced-order parallel Kalman Filters, while the second, based on the Information Theory, is made up of a set of reduced-order Information Filters. A comparative analysis of both the computational complexity and performance is faced in this chapter to determine which algorithm is better under what condition.

9.1 Algorithms Performance Comparison

Here an analysis to investigate both advantages and weakness of the proposed algorithms is proposed. Note that, both the Interlaced Extended Kaman Filter and the Extended Information Filter are derived from the Bayesian Frameworks. Therefore, it is reasonable to expect similar performance.

The evaluation is performed in terms of accuracy of the estimation, velocity of convergence and number of messages exchanged at each iteration. The sensor network deployment shown in Fig. 9.1, exploited for the analysis of both algorithms in the previous chapters, has been considered as common test-bed. Table 9.1 shows the results for both algorithms with respect to several anchors arrangements.

9.1.1 Performance Evaluation

Providing an accurate estimation of node locations is fundamental for several applications such as environmental monitoring or tracking in order to operate properly. Moreover, the localization process should be performed minimizing the computational effort. According to the table, both approaches provide satisfactory results: the estimation is accurate and it is achieved within few iterations.

As far as performances as concerned, little differences can be noticed, even though the IEKF seems to perform slightly better in terms of estimation accuracy. This can be explained by the fact that while the cross-correlation terms

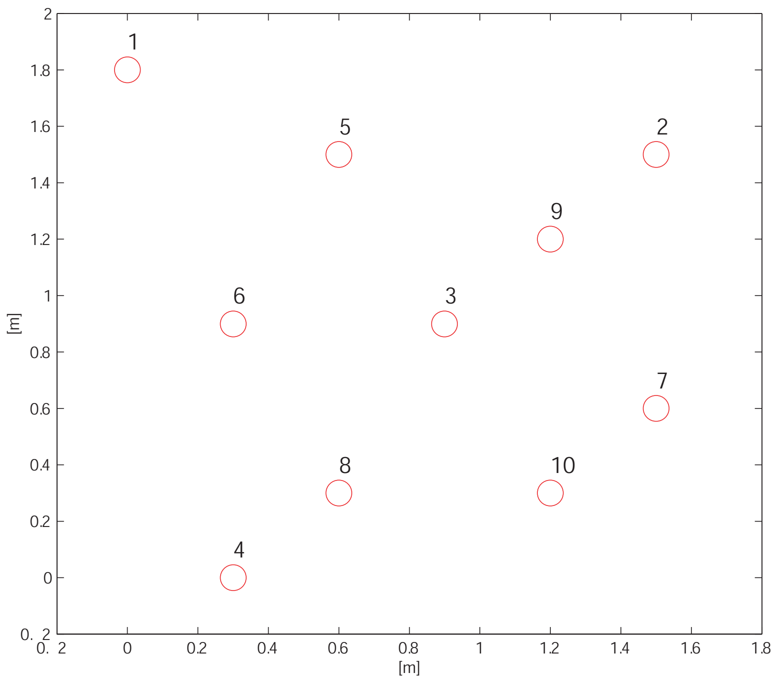

Figure 9.1: Test-bed Deployment for algorithms comparison.

Distributed Interlaced Extended Kalman Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	11.18	4.09	7.96	25
{1,6,7}	11.06	2.57	6.59	26
{5,9,10}	12.71	1.96	5.63	24

Distributed Extended Information Filter				
Anchors	Max Error [cm]	Min Error [cm]	Mean Error [cm]	Conv. [Steps]
{1,2,3}	13.13	4.28	8.70	24
{1,6,7}	13.56	1.86	7.21	33
{5,9,10}	13.44	1.93	5.64	21

Table 9.1: Distr. IEKF vs Distr. EIF.

are completely neglected by the IEF, these terms are exploited by the IEKF to suitably increase the measurement noise covariance matrix R. Therefore, the IEKF offers the advantage to partially compensate the error introduced by the de-coupling action. Conversely, as far as the velocity of convergence is concerned, both algorithms provide comparable performances.

9.1.2 Messages Exchanged

An important aspect of the algorithms evaluation is the number of messages which need to be exchanged among nodes at each iteration. The limited battery life typical of a sensor network device demands to regulate the communication among nodes in order to extend the operability of the whole network. Also in this case, both approaches seem to meet the requirement as the convergence is achieved exchanging few messages at each iteration. In detail, messages can be classified in two groups: communication and observation messages. The first type of message is used to exchange information among nodes while the second type is exploited by the ranging technique given in Appendix D. Since the ranging technique can be always substituted, the analysis is only focused onto the communication messages.

For the IEKF, at each iteration any node i within the observation range of a given node j must broadcast its latest information, i.e., $x^{(i)}_{k|k-1}$, $P^{(i)}_{k|k-1}$ These information will be exploited by the node j during the correction step to build the Kalman gain. Therefore, for every single node one message needs to be sent and M messages are supposed to be received at each iteration.

Similar is the situation for the IEF. In fact, also in this case every single node needs to be aware of the latest estimates of its neighbors in order to perform the estimation step. However, in this case only the state estimate $x^{(i)}_{k|k-1}$ must be sent.

9.2 Computational Complexity Analysis

Here, a comparative analysis regarding the computational complexity of both the Extended Information Filter and the Interlaced Kalman Filter previously proposed is faced. In order to achieve that, the asymptotic notation (a mathematical notation used to describe the asymptotic behavior of functions) is considered. Its purpose is to characterize a function's behavior for very large (or very small) inputs in a simple but rigorous way that enables comparison to other functions [11]. Furthermore, in order to easily analyze the filter equations, a formalism has been introduced with the aim of describing the matrix operations and the related computational complexity:

- `SUM(NxM,NxM)` $= \mathcal{O}(N \cdot M)$
- `SUB(NxM,NxM)` $= \mathcal{O}(N \cdot M)$

- `MUL(NxM,MxP)` $= \mathcal{O}(N \cdot M \cdot P)$
- `INV(NxN)` $= \mathcal{O}(N^3)$

Note that, for sake of simplicity, the asymptotic complexity assumed for these operations does not reflect the most efficient implementation available so far. However, it does not affect the validity of the analysis since the complexity of the most efficient implementations scale approximatively the same. Furthermore, all the elementary operations related to scalar values have been assumed with complexity $\mathcal{O}(1)$.

9.2.1 The Interlaced Extended Kalman Filter

The complexity of the Interlaced Extended Kalman Filter described in Section 7.2 running on-board of a node can be summarized as in Table 9.2.

Table 9.2: Interlaced Extended Kalman Filter Computational Load

$\hat{x}_{k\|k-1} = \hat{x}_{k-1\|k-1}$	-
$P_{k\|k-1} = [P_{k-1\|k-1} + Q_k]$	`SUM(NxN,NxN)`
$\hat{x}_{k\|k} = x_{k\|k-1} + K_k \nu_k$	`SUM(Nx1,Nx1)` `MUL(NxM,Mx1)`
$\nu_k = z_k - h(\hat{x}_{k\|k-1})$	`SUB(Mx1,Mx1)`
$K_k = P_{k\|k-1} Jh_k^T S_k^{-1}$	`MUL(NxN,NxM)` `MUL(NxM,MxM)` `INV(MxM)`
$S_k = Jh_k P_{k\|k-1} Jh_k^T + R_k$	`MUL(MxN,NxM)` `SUM(MxM,MxM)`
$P_{k\|k} = (I - K_k Jh_k) P_{k\|k-1}$	`MUL(NxM,MxN)` `SUB(NxN,NxN)` `MUL(NxN,NxN)`
$\tilde{R}_k = diag\{R_k + JH_k^j P_{k\|k-1}^j JH_k^{j^T}\}$ $\forall j \in \mathcal{N}(i)$	`MUL(1xN,NxN)` `MUL(1xN,Nx1)` `M times`

Three remarks are now in order:

- Only matrix operations have been taken into account,
- The complexity of the Jacobian construction has been neglected,
- The complexity of the observation evaluation has been neglected.

The first observation underlines that the asymptotic behavior of the algorithm is desired. The second observation comes from the consideration that the computational complexity of the Jacobian is always lighter compared to other operations.

Thus, it will be omitted for sake of clarity. The third observation follows the same reasoning as the second one.

9.2.2 The Extended Information Filter

The complexity of the distributed Extended Information Filter described in 8.2 running on-board of a node can be summarized as in Table 9.3.

Table 9.3: Extended Information Filter Computational Load

$Y_{k\|k-1} = [Y^{-1}_{k-1\|k-1} + Q_k]^{-1}$	`INV(NxN)`
	`SUM(NxN,NxN)`
	`INV(NxN)`

$L_{k\|k-1} = Y_{k\|k-1} Y^{-1}_{k-1\|k-1}$	`MUL(NxN,NxN)`
$y_{k\|k-1} = L_{k\|k-1} \hat{y}_{k-1\|k-1}$	`MUL(NxN,Nx1)`
$Y_{k\|k} = Y_{k\|k-1} + I_k$	`SUM(NxN,NxN)`
$I_k = Jh_k^T R_k^{-1} Jh_k$	`INV(MxM)` (diag) `MUL(NxM,MxM)` (diag) `MUL(NxM,MxN)`
$y_{k\|k} = y_{k\|k-1} + i_k$	`SUM(Nx1,Nx1)`
$i_k = Jh_k^T R^{-1} z'_k$	`MUL(NxM,Mx1)`

$z'_k = \nu_k + h(\hat{x}_{k\|k-1})$	`MUL(MxN,Nx1)`
	`SUM(Mx1,Mx1)`
	`INV(NxN)`
	`MUL(NxN,Nx1)`

$\nu_k = z_k - h(\hat{x}_{k\|k-1})$	`SUB(Mx1,Mx1)`

The same considerations that have been done for the IEKF still hold here.

9.2.3 IEKF vs. EIF

In order to find out the differences between the two algorithms, the matrix operations have been compared:
Table 9.4, which summarizes the set of operations required by both algorithms at each iteration, can be simplified considering that from an asymptotical standpoint, some operations, such as sum, subtraction or transposition, have a lower order than other ones, such as multiplication or inversion.

Table 9.5 can be further simplified considering that from an asymptotical point of view, the number of occurrences, if not related to any of the parameters of interest, does not influence the complexity of the algorithm.

Table 9.6 describes the subset of operations characterizing the computational complexity of the two approaches. The dominant operation for the EIF can be either the multiplication of a matrix $N \times M$ with a matrix $M \times N$ with complexity $\mathcal{O}(N^2 \cdot M)$ or the inversion of a matrix $N \times N$ with complexity $\mathcal{O}(N^3)$,

Table 9.4: Computational Complexity: Comparative Table I

IEKF	EIF
`SUM(NxN,NxN)`	`INV(NxN)`
`SUM(Nx1,Nx1)`	`SUM(NxN,NxN)`
`MUL(NxM,Mx1)`	`INV(NxN)`
`SUB(Mx1,Mx1)`	`MUL(NxN,NxN)`
`MUL(NxN,NxM)`	`MUL(NxN,Nx1)`
`MUL(NxM,MxM)`	`SUM(NxN,NxN)`
`INV(MxM)`	`INV(MxM)` (diag)
`MUL(MxN,NxM)`	`MUL(NxM,MxM)` (diag)
`SUM(MxM,MxM)`	`MUL(NxM,MxN)`
`MUL(NxM,MxN)`	`SUM(Nx1,Nx1)`
`SUB(NxN,NxN)`	`MUL(NxM,Mx1)`
`MUL(NxN,NxN)`	`MUL(MxN,Nx1)`
`MUL(1xN,NxN)xM`	`SUM(Mx1,Mx1)`
`MUL(1xN,Nx1)xM`	`INV(NxN)`
	`MUL(NxN,Nx1)`
	`SUB(Mx1,Mx1)`

Table 9.5: Computational Complexity: Comparative Table II

IEKF	A.C.C.	EIF	A.C.C.
		`INV(NxN)`	$\mathcal{O}(N^3)$
`MUL(NxM,Mx1)`	$\mathcal{O}(N \cdot M)$	`INV(NxN)`	$\mathcal{O}(N^3)$
		`MUL(NxN,NxN)`	$\mathcal{O}(N^3)$
`MUL(NxN,NxM)`	$\mathcal{O}(N^2 \cdot M)$	`MUL(NxN,Nx1)`	$\mathcal{O}(N^2)$
`MUL(NxM,MxM)`	$\mathcal{O}(N \cdot M^2)$		
`INV(MxM)`	$\mathcal{O}(M^3)$	`INV(MxM)` (diag)	$\mathcal{O}(M)$
`MUL(MxN,NxM)`	$\mathcal{O}(N \cdot M^2)$	`MUL(NxM,MxM)` (diag)	$\mathcal{O}(N \cdot M)$
		`MUL(NxM,MxN)`	$\mathcal{O}(N^2 \cdot M)$
`MUL(NxM,MxN)`	$\mathcal{O}(N^2 \cdot M)$		
		`MUL(NxM,Mx1)`	$\mathcal{O}(N \cdot M)$
`MUL(NxN,NxN)`	$\mathcal{O}(N^3)$	`MUL(MxN,Nx1)`	$\mathcal{O}(M \cdot N)$
`MUL(1xN,NxN)xM`	$\mathcal{O}(N^2 \cdot M)$		
`MUL(1xN,Nx1)xM`	$\mathcal{O}(N \cdot M)$	`INV(NxN)`	$\mathcal{O}(N^3)$
		`MUL(NxN,Nx1)`	$\mathcal{O}(N^2)$

Table 9.6: Computational Complexity: Comparative Table III

IEKF	A.C.C.	EIF	A.C.C.
`MUL(NxM,Mx1)`	$\mathcal{O}(N \cdot M)$	`INV(NxN)`	$\mathcal{O}(N^3)$
		`MUL(NxN,NxN)`	$\mathcal{O}(N^3)$
`MUL(NxN,NxM)`	$\mathcal{O}(N^2 \cdot M)$	`MUL(NxN,Nx1)`	$\mathcal{O}(N^2)$
`INV(MxM)`	$\mathcal{O}(M^3)$		
`MUL(MxN,NxM)`	$\mathcal{O}(N \cdot M^2)$		
		`MUL(NxM,MxN)`	$\mathcal{O}(N^2 \cdot M)$
`MUL(NxN,NxN)`	$\mathcal{O}(N^3)$	`MUL(MxN,Nx1)`	$\mathcal{O}(M \cdot N)$
		`MUL(NxN,Nx1)`	$\mathcal{O}(N^2)$

where N is the dimesion of the state space and M is the number of observations. Conversely, for the IEKF the dominant operation can be either the inversion of a matrix $N \times N$ with complexity $\mathcal{O}(N^3)$ or the inversion of a matrix $M \times M$ with complexity $\mathcal{O}(M^3)$.

The use of one technique over the other depends upon the reciprocal dimension between the state space and the observations. If the dimension of the state space is lower than the dimension of the observations $N < M$, the EIF turns out to be computationally more efficient than the IEKF. Conversely, if the dimension of the state space is higher than the dimension of the observations $N > M$, the IEKF performs better even though the complexity is the same from an asymptotical point of view. Indeed, this is due to the fact that several operations with cubic complexity in N are required by the EIF at each iteration. Note that for the Sensor Network scenario, the dimension of the state space for each node is fixed to $N = 2$, while the dimension of the observations is strictly related to the number of nodes Ω deployed into the environment. Therefore the Extended Information Filter turns out to be more effective than the Interlaced Extended Kalman Filter.

Special case: single observation update

Thus far an analysis where M observations were processed all together at each iteration has been provided. If a slight decrease in accuracy of the estimation is acceptable, the computational complexity can be even further reduced. This can be achieved by updating the estimate considering a single observation at a time. In this case, for the IEKF the inversion of the innovation is reduced to the inversion of a scalar. The dominant operation is given by the multiplication required for the computation of this scalar and its complexity becomes linear with the dimension of the state. However, since it has to be repeated M times the real complexity becomes $\mathcal{O}(N \cdot M)$, which is indeed significantly lower compared to the previous one ($\mathcal{O}(M^3)$). Note that, the situation for the EIF is completely different . In fact, even if the computational load required for the construction of the Innovation matrix becomes linear with the dimension of the state, several inversions of matrixes $N \times N$ are still required at each iteration. Therefore in this case any potential advantage simply vanishes.

9.3 Considerations

In the previous two chapters two novel approaches to deal with the Self-Localization Problem in Sensor Networks have been proposed. Here, these approaches, namely the Interlaced Extended Kalman Filter and the distributed Extended Information Filter, have been compared. The comparison has been performed by exploiting a common sensor network deployment, shown in Fig. 9.1, which involves ten nodes. Experimental results evidence comparable performance underlining the algebraic equivalence of the two approaches. Indeed, both approaches come from the Bayesian framework as they are derived from the "classical" Kalman Filter formulation. However, few interesting peculiarities can be pointed out.

The Interlaced Extended Kalman Filter seems to perform slightly better in terms of estimation accuracy compared to the Extended Information Filter. This can be explained by the mechanism provided by the IEKF to (partially) balance the approximation introduced by the de-coupling operation, i.e., the augmentation of the observation noise covariance matrix by the cross-correlation terms. On the other hand, the EIF seems computationally more convenient. In fact, it provides approximatively the same computational load as the the IEKF in the special case but processing all the observation together at each iteration.

Due to the algebraic equivalence of the two approaches, future research might be focused on providing such an interlacement-like strategy for the EIF as well. In this way the higher accuracy obtained by the IEKF might be achieved by the EIF too while preserving the computational advantages which make the EIF appealing in a multi-sensor fusion context.

Part III

Mobile Robotics and Sensor Networks as Integrated Framework: A case study

Chapter 10

A Framework for Multi-Robot Node Coverage in Sensor Networks

Area coverage is a well-known problem in robotics. Extensive research has been conducted for the single robot coverage problem in the past decades. More recently, the research community has focused its attention on formulations where multiple robots are considered. In this work, a new formulation of the multi-robot coverage problem is proposed. The novelty of this work is the introduction of a sensor network, which cooperates with the team of robots in order to provide coordination. The sensor network, taking advantage of its distributed nature, is responsible for both the construction of the path and for guiding the robots. The coverage of the environment is achieved by guaranteeing the reachability of the sensor nodes by the robots. Two distributed algorithms for path construction are discussed. The first aims to speed up the construction process exploiting a concurrent approach. The second aims to provide an underlying structure for the paths by building a Hamiltonian path and then partitioning it. A statistical analysis has been performed to show the effectiveness of the proposed algorithms. In particular, three different indexes of quality, namely completeness, fairness, and robustness, have been studied.

10.1 Introduction

Area coverage has been investigated by the robotics research community through the years. Indeed, this problem lends itself to several applications in different fields, from industrial, such as lawn-mowing [5] or vacuum-cleaning [50], to military such as de-mining [69], or humanitarian, such as search and rescue operations [125].

The robot area coverage problem is the problem of determining a path that must be followed by a robot in order to completely cover the environment. Several approaches, ranging from grid decompositions of the environment to the development of heuristics, have been proposed. More recently, formulations which extend the problem to the multi-robot context have been introduced. The idea is to take advantage of the cooperation among the robots to provide higher robustness as well as to lower the time required to complete the task.

Indeed, three major issues define the area coverage problem, according to [46]: the capability to generate paths that are able to completely cover the environment; the time required to complete the coverage operations; and finally, the availability of a priori information of the environment. Additionally, a fourth issue derives from the presence (or absence) of obstacles.

In this work,, a new formulation for the Multi-Robot Coverage problem is proposed. The novelty of this work is the introduction of a sensor network, which cooperates with the team of robots in order to provide coordination to it. This work investigates how the dynamics of the area coverage problem are modified when introducing a sensor network into the system. The sensor network, taking advantage of its distributed nature, is responsible for both the construction of the path and for guiding of the robots. Here, we focus on the distributed construction of paths. Specifically, two distributed algorithms have been provided. The first one aims to speed up the construction process exploiting a concurrent approach. The second aims to provide an underlying structure by building a Hamiltonian path and then partitioning it. A statistical analysis has been performed to show the effectiveness of the proposed solutions. In particular, three different indexes of quality, namely completeness, fairness, and robustness, have been studied.

10.2 Related Work

10.2.1 Robot Area Coverage

The (area) coverage problem was shown to be related to the covering salesman problem in [6]. Specifically, the covering salesman problem is a variant of the traveling salesman problem where, instead of visiting each city, an agent must visit a neighborhood of each city that minimizes the travel length for the agent. In [6], this problem was proven to be NP-hard making use of the reduction from the (NP-hard) problem "Hamiltonian Circuit in Planar Bipartite Graphs with Maximum Degree 3" to the problem "Hamiltonian Circuit in Grid Graphs".

In [46], two different formulations of the coverage problem are described: *offline* and *online*. The offline formulation assumes robots to be equipped with a map of the work area, [68, 79, 2]. The online formulation does not assume any prior information about the environment to be available for the robots, [93, 143, 131]. Moreover, coverage algorithms can be classified into deterministic and non-deterministics according to [128]. Deterministic approaches ([95, 111, 40, 161]) guarantee the complete coverage of the environment, while non-deterministic approaches ([151, 17, 44]) cannot.

In [68], the problem of covering a continuous planar area by a square-shaped tool attached to a mobile robot is addressed. Here, the authors suggest to sub-divide the work-area into disjoint cells, whose size is related to the tool carried by the robot. Successively, a spanning tree of the graph induced by the cells is performed, while covering every point precisely once. Hence, the coverage of the environment is simply obtained by letting the robot circumnavigate the tree.

In [79], an offline formulation of the multi-robot coverage problem is addressed. A path-planning algorithm which extends the idea proposed in [68] to the multi-robot scenario is described. The main contribution of this work is the introduction of the Multi Spanning Tree Coverage (MSTC) problem formulation. The idea is to optimally divide the spanning-tree previously computed in such a way that each robot covers an equal portion of the tree. An argumentation about the robustness and the efficiency of the proposed algorithm is provided as well.

In [2], an extension of the previous approach is suggested. This work is motivated by the observation that the coverage time can be significantly influenced by the structure of the spanning trees. As a result of their investigation, the authors claim that an optimal time coverage algorithm for a system with k robots will result (at least theoretically) in total coverage time of $\lceil N/k \rceil$, where N is the number of cells. In addition, they prove that robots should be spread out as uniformly as possible along the spanning tree in order to achieve this goal. In fact, in this way the k paths will result in almost the same length.

In [93], the authors propose an algorithmic approach to deal with the distributed complete coverage problem. This work represents an extension to the single robot sensor-based coverage of unknown environments proposed in [1]. By assuming that global communication is available among robots, each robot is assigned an area of the unknown environment to cover. This area is decomposed into cells and it is described by an adjacency graph which is incrementally built. In detail, the Morse decomposition based on the Boustrophedon approach, first proposed in [47], is exploited by each robot. Robots can achieve a global picture of the environment by integrating each graph with information coming from other robots.

In [143], an algorithm for the complete multi-robot coverage of a connected space with unknown obstacles is presented. This algorithm operates by maintaining, as far as possible, small uncovered regions between covered areas and obstacles. In addition, the authors show that repeated coverage can occur only around regions where the paths between obstacles are less than twice the width of the robots coverage range. This property holds even when the robots have no *a priori* knowledge of the environment, and therefore helps to prevent unnecessary wastage of time and resources.

In [131], the authors introduce a novel strategy for the exploration of an unknown environment with a multi-robot system. Communication among robots is restricted to line-of-sight and to a maximum interdistance between robots. The proposed strategy produces a spring-like formation which scan the area in strips. Additionally, in the presence of obstacles the formation is deformed and split in two in order to circumvent the obstacle and to adapt to the varying width of the free space. Moreover, the authors provide an upper bound for the amount of repeated coverage. This is limited to the areas where paths between obstacles are too narrow to allow robots to enter and leave the area on non-overlapping paths.

In [95], an algorithm for efficient cooperative search is described. The idea is to divide a work area into small pieces in order to make multiple mobile robots cooperate efficiently. The searching motion of mobile robots is represented by a queue of paths and the division of the work areas is achieved by allocating appropriate paths to each robot respectively. In this way, the cost of the searching can be shared among the multiple mobile robots.

In [111], the problem of cooperative area sweeping by multiple mobile robots is addressed. The authors propose a non path-following approach called On-Line Goal Selection (OGS) algorithm for robot motion planning in autonomous behavior. Cooperation is achieved by organizing the robots as a decentralized market-like structure and task-sharing is addressed by exploiting a negotiation mechanism.

In [40], a distributed cooperative coverage algorithm named DC_R is described. It represents an extension of an earlier complete single-robot algorithm called CC_R [39]. In detail, DC_R executes independently on each robot in a team where the individual robots do not know the initial locations of their peers and applies to systems of robots operating in a rectilinear environment that use only intrinsic contact sensing to determine the boundaries of the environment.

In [161], a polynomial-time multi-robot coverage heuristic named the Multi-Robot Forest Coverage (MFC) is proposed. This approach relies on an algorithm for finding a tree cover with trees of balanced weights (one for each robot). In addition, the authors provide an analysis which shows the cover time to be at most eight times larger than the optimal one.

In [151], the authors introduce a distributed algorithm exploiting the indirect form of communication adopted by ants. Robots leave chemical odor traces which evaporate over time and evaluate the strength of smell at every point they reach, with some measurement error. The effectiveness of this communication scheme to perform the task of cleaning the floor of an unmapped building, or more broadly any other task requiring the traversal of an unknown region is investigated.

In [17], two algorithms for solving the 2D coverage problem using a team of robots are described. Specifically, these algorithms rely on the observation that local dispersion is a natural way to achieve global coverage. Therefore, both algorithms are based on local, mutually dispersive interaction between robots when they are within sensing range of each other. In fact, "spreading out" robots throughout the environment lowers the risk of having robots too close to each other, which would result in poor coverage as overlap would be experienced.

In [44], a biologically inspired neural network approach to autonomous cooperative coverage path planning of multiple cleaning robots is proposed. Paths are generated on-line by using a neural network, where the dynamic of each neuron is characterized by a shunting neural equation. Robots see each other as moving obstacles and cooperate to achieve a common sweeping goal.

10.2.2 Hamiltonian Path

In the mathematical field of graph theory, a Hamiltonian path is a path in an undirected graph which visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex. Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem which is NP-complete [13].

The Hamiltonian path problem for graph G is equivalent to the Hamiltonian cycle problem in a graph H obtained from G by adding a new vertex and connecting it to all vertexes of G. The Hamiltonian cycle problem is a special case of the traveling salesman problem (TSP), obtained by setting the distance between two cities to a finite constant if they are adjacent or to infinity otherwise.

Several centralized approaches providing either exact or approximate solutions for the TSP have been proposed in literature. Exact solutions are generally obtained exploiting branch and bound techniques [149], linear programming formulations [53], or integer programming formulations [109]. Approximate solutions are achieved using heuristic local search methods. Heuristics are largely applied to solve large instances of the TSP as they can compute near optimal solutions in a relatively short time. Starting from the classical tour construction heuristics such as nearest neighbor heuristics [87], insertion heuristics [132], heuristics based on spanning trees [48], savings heuristics [49], and 3-opt [24], further heuristic approaches have been proposed like simulated annealing [91], genetic algorithms [116], neural networks [32] or ant-colony [56].

Distributed implementation methods can be found in literature as well. For instance, a distributed implementation of simulated annealing is described in [4], while a distributed implementation of a parallel genetic algorithm on a cluster of workstations is given in [138]. Another distributed approach, which provides a parallelization of a branch-and-bound algorithm, is proposed in [147].

Many variations of the classical TSP formulation have been proposed over the years. Among them, the Multiple Traveling Salesman Problem (MTSP) is the closest to the Multi-Robot Coverage Problem. An interesting extension of the MTSP is the balanced version for which all paths are required to be of equal length. An overview of the MTSP formulation and solution procedures can be found in [22].

10.3 Problem Formulation

Given a sensor network described by a graph $G = (N, L)$ where N is the set of nodes (sensors) with cardinality n and L is the set of edges (connectivity matrix) and, given a set K of robots with cardinality k, the following assumptions are made:

- The sensor network is deployed within an open environment without obstacles. (The deployment might be either manual or automatic by a robot. If by robot, the algorithm proposed in [18] or [19] might be used),
- The *quasi-uniform* deployment is obtained by placing each sensor node at exactly one vertex of a noisy grid built over the whole environment,
- Each node knows its location in respect to a global frame (the localization can be achieved for instance using one of the two approaches proposed in Part II),
- There is a link layer that provides the abstraction of reliable communication (for instance using re-transmission),
- The cardinality of the robots set is strictly lower than the cardinality of the sensors set: $k << n$.

According to this scenario, the coverage problem consists of constructing k paths, where a path is defined as an open succession of 2-connected nodes, so that:

- each node belongs to a path
- any pair of paths is disjointed
- all paths are of equal length (optimality condition)

Hence, the coverage of an environmet is achieved by guaranteeing the reachability of the nodes by the robots. Note that, the introduction of a sensor network allows some assumptions to be relaxed. Specifically, the following significant differences can be pointed out with respect to the "off-line coverage" formulation:

- No prior knowledge of the environment is required for the robots
- The number of robots, either joining or leaving the network, can be assumed to be dynamic

At the same time, this formulation leads to new interesting questions:

- How to provide a fully distributed framework able to build paths as close to optimality as possible?
- How to handle the eventuality that a robot might join (respectively leave) the network. This would imply the ability of the network to re-organize the pre-existing paths in order to keep the "optimality" condition.

This formulation can be described by the Integer Linear Programming formulation proposed in Appendix E.1 if the distance between adjacent nodes is considered approximately constant, as a consequence of the *quasi-uniform* deployment. In this case, any feasible solution is inherently optimal. However, it is legitimate to ask how close to optimality can a solution be if a distributed formulation is considered. Therefore, the following properties, particularly the *fairness* of the solution, are investigated in order to evaluate the effectiveness of the proposed solution:

- *Completeness* in terms of network coverage, which means having each node belonging to a path
- *Fairness* in terms of equal distribution of the duty among the robots, which means having paths in average of the same length
- *Robustness* in terms of ability to re-cover from the situation in which a robot might get broken or a new one might join the network

Indeed, these indexes were already adopted in [79]. Although some similarities might be pointed out, this work is mainly focused on investigating how the dynamics of the problem are modified by the deployment of a sensor network within the environment.

10.4 The proposed solution

Two different distributed approaches are proposed. Both of them have strengths and weaknesses that will be discussed in detail. However, due to the NP-hard nature of the problem, none of them is able to guarantee optimality in a deterministic way. For this reason, statistical analysis is provided in order to validate their effectiveness. Following the formulation given in Section 10.3, it can be observed that a fair distribution of the duty is achieved if the number of nodes belonging to any path is approximately $\lceil N/k \rceil$. Indeed, this is in agreement with the condition of optimality proposed in [2], for which an optimal algorithm should lead to a total time coverage of $\lceil N/k \rceil$. In fact, due to the *quasi-uniform* deployment of the sensor network, the time required for a robot to travel between two neighbors can be approximatively considered a constant c in average. Therefore, a path can be traversed in $c \cdot \lceil N/k \rceil$.

10.4.1 Algorithm I: Overview

The first algorithm constructs paths in a concurrent way. The idea is to use the distributed nature of the sensor network to speed up the construction process. There are tree steps:

- Heads Selection
- Coarse Paths Construction

- Orphan Recovery Policy

Heads Selection

When the presence of the robots is sensed, this information is spread out across the network, and the heads selection process is triggered. The idea is to select heads, i.e., nodes where the path construction process is started from, so that a natural partition of the environment is achieved. Intuitively, this can be explained by the fact that, as the algorithm is distributed and the construction of the paths is concurrent, minimizing the interaction improves the performance. More details will be provided in the next section.

Coarse path construction

This process is started as soon as the heads have been chosen. As previously stated, the idea is to exploit the distributed nature of the network to speed up the path construction process. Several policies to make the more effective local decision have been investigated and their peculiarities will be shown in the next section. The pseudo-code in Algorithm (5) shows a possible distributed implementation of the proposed solution, where v and p are respectively the current tail of the path and its predecessor, $\mathcal{N}(i)$ is the set of neighbors of node i, $\mathcal{N}_{av}(i) \subset \mathcal{N}(i)$ is the subset of available neighbors describing the possible candidates to be added and $d(i, j)$ represents a distance between node i and node j. Fig. 10.1 shows a typical result of the coarse path construction procedure.

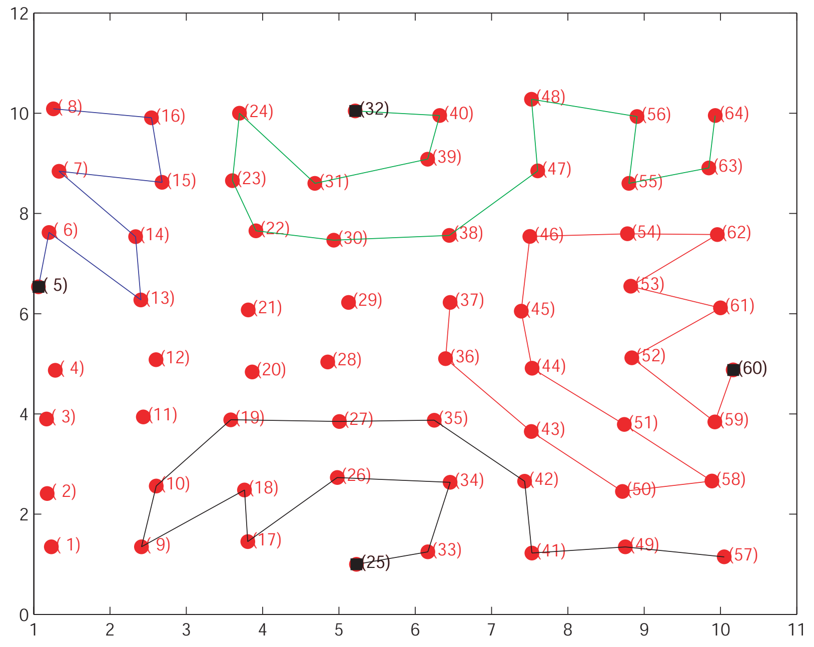

Figure 10.1: Coarse path construction

Algorithm 5: Coarse path construction

```
1  $v \leftarrow$ Tail
2  $p \leftarrow$ Tail's predecessor
3  $v$ broadcasts request of availability to $\mathcal{N}(v)$
4  $\mathcal{N}_{av}(v) \leftarrow v$ gains availability responses
5  $d \leftarrow \infty$
6  $n \leftarrow 0$
7  for $j \in \mathcal{N}_{av}(v)$ do
8      $v$ computes $d(v, j)$
9      if $d(v, j) < d$ then
10         $d = d(v, j)$
11         $n = j$
12     end
13 end
14 if $n = 0$ then
15     $v$ notifies to $p$ the end of construction process
16 else
17     $v$ broadcasts the request of attachment of $n$
18     $v$ receives the reply $r$ from $n$
19     if $r = success$ then
20         $v$ notifies to $n$ to start the path construction process
21     else
22         $N_{av}(v) = N_{av}(v) \setminus \{n\}$
23         Go to 5
24     end
25 end
```

Orphan Recovery Policy

Due to the distributed nature of the system as well as the locality of the decision process, a second step is required in order to fully cover the environment. Once the first step is terminated, there might be several incomplete paths with different lengths. For this reason, a local shortest-path-first-served mechanism is provided in order to balance the final length of the paths.

Nodes which have not been added to any path, after a period of latency of attachment request put themselves in the "orphan" status and start this process. Note that, particular attention should be paid to avoid the formation of crosslinks when adding orphans to the paths. For this reason, an ad-hoc policy to avoid this issue has been devised and will be discussed in the next section. The pseudo-code in Algorithm (6) shows a possible distributed implementation of this step, where v is an orphan, $M(i, j)$ is the crossing links check function, $Att(i, j)$ is the checks . Fig. 10.2 depicts how the paths are modified after the Orphan Recovery Policy is run.

10.4.2 Algorithm II: Overview

The second algorithm involves the construction of an approximate Hamiltonian path in a distributed way. The idea is to first build a continuous "backbone" and successively provide an optimal partition of it. The proposed solution involves two steps:

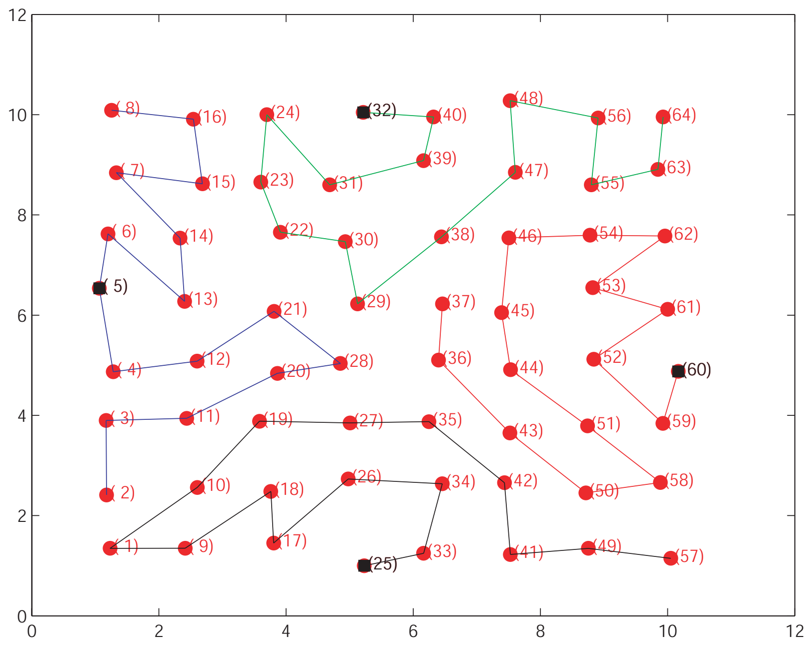

Figure 10.2: Final Paths

- Approximate Hamiltonian Path Construction
- Path Partitioning

This algorithm, similar to the “Route First-Cluster Second” approach proposed in [21] for transportation scheduling, has the great advantage of providing an underlying structure for the obtained paths. This property turns out to be very interesting when investigating the robustness of the algorithm(considering a robot either joining or leaving the scene). In fact, the availability of a continuos “backbone” allows for an easy reconfiguration, i.e., modification of paths, by exploiting only local information.

Approximate Hamiltonian Path Construction

When the presence of the robots is sensed, the approximate Hamiltonian path construction process is triggered. A fully distributed algorithm which makes local decisions based on a heuristic approach is provided. Moreover, an additional step, namely the “Path Refining” process, is given as well in order to add nodes which have not been included into the path after the first step is run. Note that, the proposed algorithm builds an approximate Hamiltonian path, since the final path might not include all nodes due to the locality of the decision process. However, these nodes can be simply added to the paths as branches. This situation does not significantly influence the performance, as in practice it happens very rarely and involves a negligible percentage of nodes (less than 1%). The pseudo-code in Algorithm (7) shows a possible distributed implementation of the first step of the

Algorithm 6: Orphan Recovery Policy

```
$v \leftarrow$ Orphan
$v$ broadcasts request of attachment to $\mathcal{N}(v)$
$\mathcal{N}_{av}(v) \leftarrow v$ gains availability responses
$\mathcal{N}_{sav}(v) \leftarrow v$ sorts $\mathcal{N}_{av}(v)$ w.r.t. ascending length of path
$M,\ M_B \leftarrow \infty$
$B,\ p,\ p_B \leftarrow 0$
for $j \in \mathcal{N}_{sav}(v)$ do
    $Att(v,j) \leftarrow v$ computes attachment condition
    if $Att(v,j) = Branch$ then
        $v$ computes $M'(v,j)$
        if $M'(v,j) < M_B$ then
            $M_B = M'(v,j)$
            $p_B = j$
        end
    else
        $v$ computes $M(v,j)$
        if $M(v,j) < M$ then
            $M = M(v,j)$
            $p = j$
        end
    end
end
if $p \neq 0$ then
    $v$ sends request of attachment to $p$
    $v$ receives the reply $r$ from $p$
    if $r = success$ then
        $v$ sets its status from “orphan” to “visited”
    else
        Go to 2
    end
else if $p_B \neq 0$ then
    $v$ sends request of attachment to $p$
    $v$ receives the reply $r$ from $p$
    if $r = success$ then
        $v$ sets its status from “orphan” to “visited”
    else
        Go to 2
    end
end
```

proposed approach, while the path refining process is the same as in Algorithm (6) without sorting the neighbors. Fig 10.3 shows a typical result of the Hamiltonian path construction procedure after the first step, while Fig 10.4 depicts the final path once the refinement has been run.

Path Partitioning

This second step aims to fairly distribute the duty among the robots so that each of them has to travel in average the same distance. The availability of a Hamiltonian path along with the knowledge of the number of robots within the network make it possible to solve this process locally without the requirement of additional communication among nodes. In detail, the i-th node performs the following operation to find out which path it belongs to:

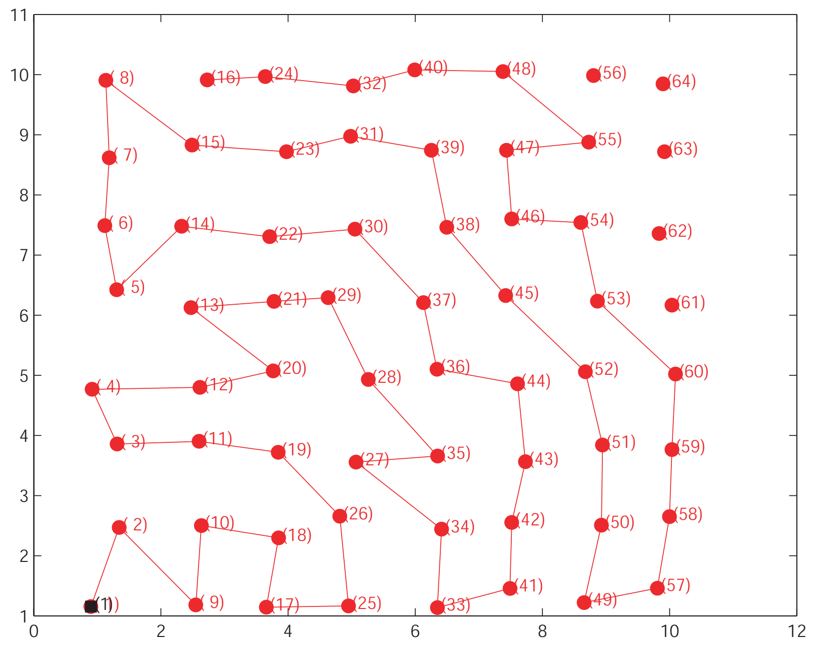
Figure 10.3: Approximate Hamiltonian Path construction

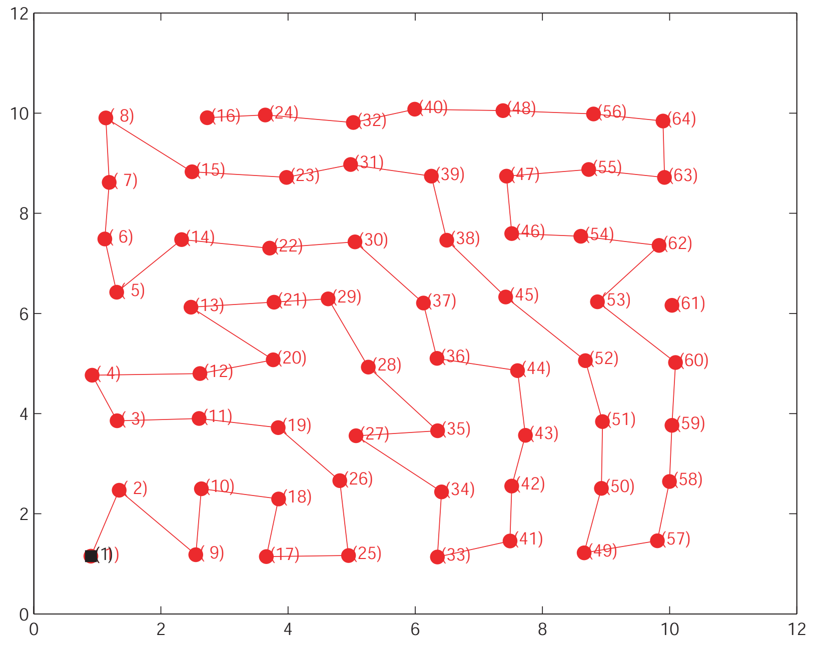
Figure 10.4: Refining Step

$$P_i = \begin{cases} \left\lceil \frac{h_i - N(\bmod\ k)\cdot\lceil\frac{N}{k}\rceil}{\lfloor\frac{N}{K}\rfloor} \right\rceil + N(\bmod\ k) & if \quad h_i > N(\bmod\ k)\cdot\lceil\frac{N}{k}\rceil \\ \left\lceil \frac{h_i}{\lceil\frac{N}{k}\rceil} \right\rceil & \text{otherwise} \end{cases} \tag{10.1}$$

Algorithm 7: Approximate Hamiltonian path construction

```
$v \leftarrow$ Tail
$p \leftarrow$ Tail's predecessor
$v$ broadcasts request of availability to $\mathcal{N}(v)$
$\mathcal{N}_{av}(v) \leftarrow v$ gains availability responses
$d \leftarrow \infty$
$n \leftarrow 0$
for $j \in \mathcal{N}_{av}(v)$ do
    $v$ computes $d(i, j)$
    if $d(v, j) < d$ then
        $d = d(v, j)$
        $n = j$
    end
end
if $n = 0$ then
    $v$ notifies to $p$ the end of the path construction
else
    $v$ broadcasts the notification of attachment of $n$
    $v$ receives the acknowledgments of notification by $n$
end
```

where, P_i is an integer representing the path number, N is the number of nodes, k is the number of robots (paths), h_i is the position of the i-th node in the Hamiltonian path and, $\lceil x \rceil$ is the ceiling function which converts the argument x to the smallest integer not less than x, while $\lfloor x \rfloor$ is the floor function which converts the argument x to the highest integer less than or equal to x. The equation (10.1) is a consequence of the following observation about the division operator:

$$a = b \cdot c + d \tag{10.2}$$

$$a = (b - d) \cdot c + d \cdot (c + 1) \tag{10.3}$$

which simply says that a set of a nodes can be partitioned in d subsets of cardinality $(c + 1)$ and $(b - d)$ subsets of cardinality c. In this way, a simple but effective way to guarantee fairness when partitioning the Hamiltonian path is achieved.

10.5 Algorithmic Analysis

The proposed algorithms are made up by several steps. In this section, a deeper investigation for the most relevant aspects is provided.

10.5.1 Heads Selection

As the system is fully distributed and the paths are built concurrently, the selection of the heads turns out to be crucial for the performance of the algorithm. Note that, the effectiveness of a selection cannot be validated independently from the other steps. Indeed, it is strictly related to the local policy that has been adopted for the path construction. This underlines the complexity that arises when dealing with distributed algorithms.

Several approaches have been investigated, the following two turned out to be the most prominent:

- Selecting the heads roughly at the center of mass of the k regions in which the environment is supposed to be partitioned
- Selecting the heads equidistantly along the border of the network

Both strategies confirm the intuition for which having a natural partition of the environment allows each path to "grow" independently. In fact, a minimization of the interactions leads to a minimization of the packets that need to be sent over the network. This is explained by the fact that the majority of the traffic over the network is due to the negotiations of a node among different paths. Simulations have been performed to prove the effectiveness of both approaches. According to the results, the second technique turns out to scale better with the size of the network. Indeed, it was quite expected, as it was developed purposely to overcome the limitations of the first technique.

10.5.2 Local Policies for Path Construction

The area coverage problem has been proved to be NP-hard [6]. The distributed nature of the system does not help to make it easier. Therefore, the assumption of considering heuristics to solve the problem is reasonable. Here, the idea is to provide a local mechanism to effectively explore the environment.

At the beginning the following simple policies have been investigated:

- Naive Random Policy
- Closest-Neighbor Policy
- Closest-to-the-Head Policy
- Closest-to-the-Barycenter Policy

As expected, the random policy does not provide good results, though it might be considered as a lower bound for the performance of the system. Unfortunately, the policy based on the choice of the closest neighbor does not produce satisfying results either. This can be explained by the fact that, none of them provides any kind of cooperation. Note that, cooperation is meant in an "implicit" way. In other words, it should be achieved as an emerging phenomenon rather than by an explicit action. This motivation inspired the development of the last two policies, which showed to perform significantly better. In fact, as paths become "attractive", interactions get implicitly minimized (emerging cooperation).

However, the best performances have been achieved when considering a *hybrid* policy. More clearly, starting from the following considerations:

- Both the Closest-To-The-Head Policy and the Closest-To-The-Barycenter Policy produce paths which do not move too far away from the head. However, the paths tend to be a little bit too jagged

- The Closest-Neighbor Policy produces paths which are very straight. However, they tend to interfere with each other.

A solution, which introduces the concept of *virtual distance*, has been devised. Specifically, a virtual distance is defined as the linear combination of two factors:

$$Vd(x_1, x_2, x_3) = \alpha \cdot \|x_1 - x_2\| + (1 - \alpha) \cdot \|x_1 - x_3\| \tag{10.4}$$

where, $\{x_1, x_2, x_3\} \in \mathcal{R}^2$ represent three location, $\|x\|$ is the Euclidean metric and α is a tunable parameter. At this point, a flexible way of taking into account two different aspects when making local decisions is provided.

10.5.3 Crossing Links Formation

The crosslink formation has been thoroughly investigated for several problems, such as geographical routing ([90, 89]). Here, a simple solution, which turns out to be very effective for the particular structure of the problem, has been adopted. In order to explain it, consider the situation depicted in Fig. 10.5. According

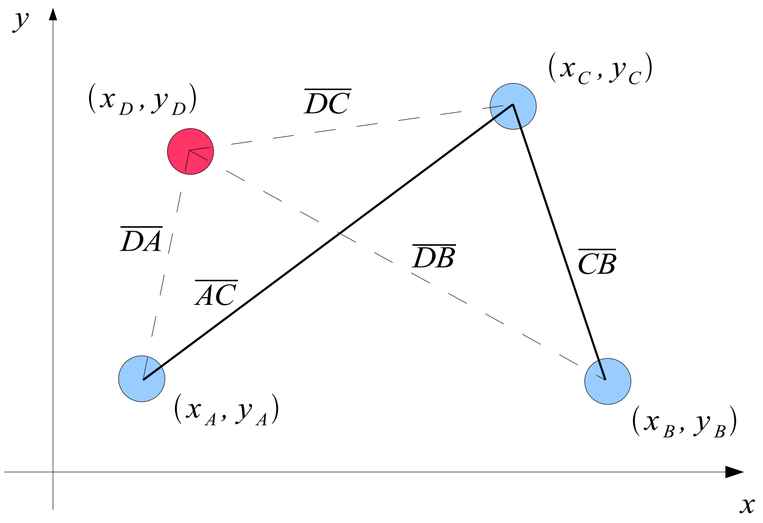

Figure 10.5: Crosslink situation

to it, nodes $\{A, B, C\}$ belong to the same path, while node D is an orphan. Moreover, the pair of segments $\overline{AC}, \overline{CB}$ represent the actual paths, while the pair of segments $\overline{DA}, \overline{DC}$ and $\overline{DC}, \overline{DB}$ describe the links that would be added if the orphan D were attached respectively to the pair of nodes $\{A, C\}$ or $\{B, C\}$. It is easy to verify that only one of the two options is "safe", namely attaching D to the pair $\{A, C\}$. The proposed solution leads to the safe decision, computing the difference between the lengths of the additional pair of links that would be added and the link that would be removed. The intuition behind this method is that the removal of long links coupled with the addition of short ones should result in lowering the chances to create a crosslink, at least from a probabilistic standpoint. In practice, in the case of the example it would be:

$$Sol = \underset{\{1,2\}}{\operatorname{argmin}} \{M_1, M_2\} \tag{10.5}$$

where

$$\begin{aligned} M_1 &= \overline{DA} + \overline{DC} - \overline{AC} \\ M_2 &= \overline{DB} + \overline{DC} - \overline{CB} \end{aligned} \tag{10.6}$$

represent the difference respectively for the first and second pair of nodes. Fig.

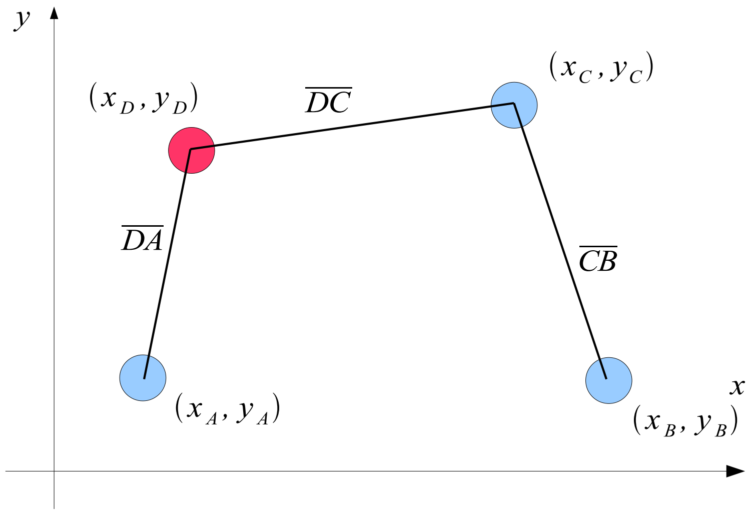

Figure 10.6: Crosslink solution

10.6 shows the correct attachment for the orphan D to the pair $\{A, C\}$.

10.6 Analysis on Completeness, Fairness & Robustness

Some properties, namely completeness, fairness and robustness, have been introduced in Section 10.3, as indexes of quality to evaluate the effectiveness of an approach. Here, an investigation w.r.t. these indexes for both algorithms is provided.

10.6.1 Test Case

In order to evaluate the performance of the proposed algorithms, a statistical analysis was performed. A simulation environment developed under Matlab by the authors has been exploited. The analysis involved several configurations with an increasing number of nodes, $(64, 132, 256)$. Each configuration were tested considering an increasing number of robots as well, $(4, 6, 8)$. The connectivity graph was obtained according to the following parameters: percentage of connected node 90%, range of connectivity 15% of the longest distance between two nodes. Finally, for each configuration, 100 trials have been run.

10.6.2 Algorithm I

This algorithm allows to build paths concurrently exploiting the distributed nature of the sensor network.

Completeness

According to the simulation results, the algorithm is always able to guarantee *completeness* as long as there are no isolated nodes. Tab. 10.1. shows the percentage of‘ “covered” nodes achieved with the first step, namely the Coarse Path Construction, and the second one, namely the Orphan Recovery Policy. Although, the algorithm cannot guarantee the full coverage of the environment

Table 10.1: Completeness: Success Percentage

Number of Nodes	Number of Robots	First Step [% Success]	Secont Step [% Success]
64	{4, 6, 8}	{82.06, 88.83, 88.75}	{100, 100, 100}
132	{4, 6, 8}	{81.44, 81.54, 81.43}	{100, 100, 100}
256	{4, 6, 8}	{86.00, 86.55, 88.10}	{100, 100, 100}

with only one step, the statistical analysis proved the network to be fully covered after the second step is run.

Fairness

The algorithm has been devised with the aim of balancing the duty among the robots. Although optimality cannot be guaranteed (in a deterministic way), statistical results, shown in Tab. 10.2, proves the effectiveness of the proposed solution. Note that, the ratio $\frac{Std}{Mean\,Value}$ is percent.

Table 10.2: Fairness

Number of Nodes	Number of Robots	Path Length Mean Value	$\frac{\text{Std}}{\text{Mean Value}}$
64	{4, 6, 8}	{16, 10.6, 8}	{10.83, 8.84, 16.54}
132	{4, 6, 8}	{33, 22, 16.5}	{6.94, 4.90, 5.25}
256	{4, 6, 8}	{64, 42.67, 32}	{5.41, 2.92, 3.49}

Robustness

This algorithm speeds up the construction of the paths by means of a concurrent approach. However, this is achieved exploiting locally a heuristic procedure. Consequently, the resulting paths do not have a common baseline. Therefore, providing a generic approach to properly modify them whether a robot leaves or joins the network, is far from being easy to realize. In practice, simulations proved that any time a robot joins or leaves the network, it is much more convenient to re-build from scratch the paths rather than trying to modify the pre-existing ones. Indeed, this is the main limitation of this algorithm, which is only partially reduced by the fact that the time required to build the k paths is significantly lowered by the concurrent approach.

10.6.3 Algorithm II

This algorithm first builds an approximate Hamiltonian path, and successively performs an optimal partition of it.

Completeness

As the algorithm builds a Hamiltonian path before to assign a duty to each robot, the completeness is structurally guaranteed as long as there are no isolated nodes. Specifically, Tab. 10.3 shows the percentage of nodes belonging to the approximate Hamiltonian path after the first and second step. In addition, the percentage of nodes attached as branches is provided as well. According

Table 10.3: Completeness: Success Percentage

Number of Nodes	First Step [% Success]	Second Step [% Success]	Isolated Branches [% Nodes]
64	85.19	100	0.11
132	85.23	100	0.08
256	82.04	100	0.07

to the statistical results, shown in Tab. 10.3, although the algorithm cannot be guaranteed to always provide the exact Hamiltonian Path, in practice it performs very well.

Note that, the problem of computing a Hamiltonian path is hard, whether the nature of the system is distributed or not. Indeed, the formulation based on the Integer Linear Programming, provided in Appendix E.1, gives an evidence of this complexity. In fact, focusing the attention on the constraint (E.7), known in literature as “Sub-tour elimination constraint” [99], it can be easily recognized that the complexity of the problem grows exponentially, in terms of number of constraints added to the problem, w.r.t. the number of nodes.

Here, although simplicity and reduced computational complexity have been taken into account when devising this algorithm, the performance achieved is satisfactory. This is due to the particular nature of the adopted connectivity graph. Nevertheless, for the context in analysis, this assumption is reasonable, as the desired connectivity graph can always be achieved by means of an appropriate tuning of the communication range of the nodes. In practice, the real requirement is merely the availability of local connectivity between the neighbor as depicted in Fig. (10.7).

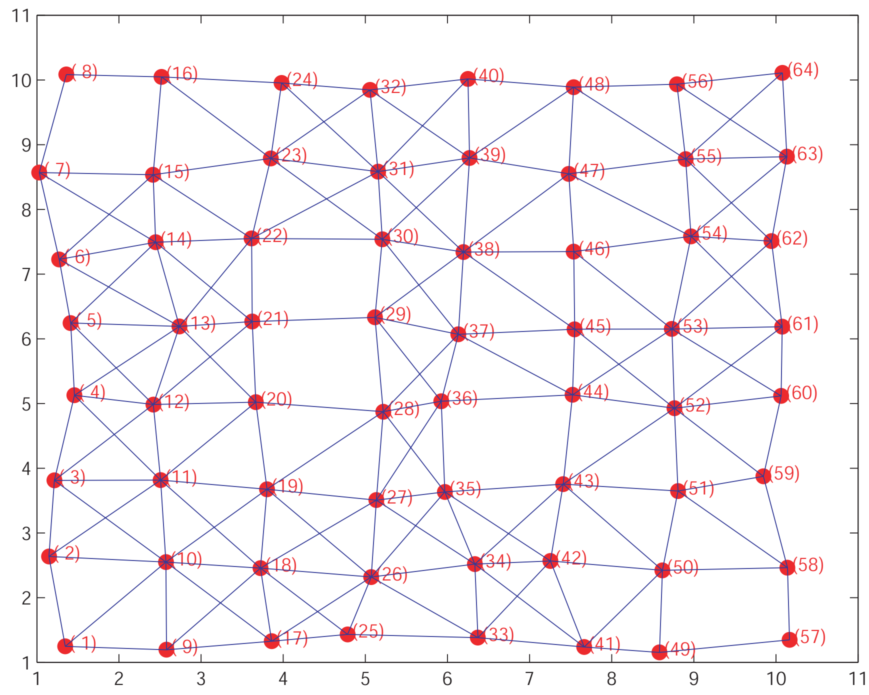

Figure 10.7: Connectivity

Fairness

The algorithm has been devised with the aim of equally distributing the duty among the robots. Moreover, unlike the previous approach, here the availability of an exact Hamiltonian path guarantees optimality in a deterministic way. However, as shown in Table 10.3, the Hamiltonian path, due to the distributed nature of the computation, might not be optimal.

Robustness

The eventuality that a robot might either get broken or join the network can be handled in a systematic way. This is due to the fact that the availability of a Hamiltonian path provides a common baseline for the assignment of the paths. Moreover,both adding a path or deleting a pre-existing one are simply a specialization of the general rule provided in 10.4.2. Note that, another interesting aspect can be pointed out: the *reactivity* of the network, i.e. the time required for the network to re-organize itself anytime a robot gets broken or a new one joins the network. In fact, as soon as one of these events is recognized, the time required for the network to re-organize itself is not related to the re-construction of the paths, which is instantaneously given by the rule (10.1), but it is only related to the notification time, i.e., the time required to spread out the information across the network. This is indeed an interesting property of the algorithm.

10.7 Performance Analysis

In this section, an analysis about the the complexity in terms of computational and communication load, of both algorithms is provided.

10.7.1 Algorithm I

Computational Load

During the "Coarse Path Construction", according to the pseudo-code proposed in Algorithm (5), the dominant operation is the loop (lines: 7 – 13) in which the candidate to be added is chosen. The worst case is achieved when the current tail has to sent p times the request of attachment, where p is the degree of a node, i.e., the number of its neighbors. The related complexity is

$$\begin{aligned} C_1(p) &= O(p) + O(p-1) + \cdots + O(1) && (10.7)\\ C_1(p) &= O(p^2) && (10.8)\end{aligned}$$

where p is the max number of neighbors of each node. Note that, the pseudo-code provided in Algorithm (5) does not reflect the most efficient implementation. In fact, a priority queue might be introduced to easily lower the complexity to $O(p)$. However, this version has been preferred for sake of clarity.

During the "Orphan Recovery" step, according to the pseudo-code proposed in Algorithm (6), the dominant operation is the loop (lines: 7 – 22) in which the orphan selects the path to be added to. The related complexity is equal to $O(p)$ where p is the max degree of any node. The worst case is achieved when the request of attachment for the orphan has to be sent p times. Thus, the complexity turns out to be

$$C_2(p) = O(p^2). \qquad (10.9)$$

Therefore, the overall computational complexity for the construction of each path is equal to:

$$\begin{aligned} C(N', p) &= N' \cdot [\alpha \cdot C_1(p) + (1-\alpha) \cdot C_2(p)] && (10.10)\\ C(N', p) &= O(N' \cdot p^2) && (10.11)\end{aligned}$$

where, $N' = \frac{N}{k}$ is the average number of nodes of a path, with N the number of nodes and k the number of robots, and α is the percentage of nodes covered at the first step, which can be retrieved by the statistical analysis.

Communication Load

During the "Coarse Path Construction", according to the pseudo-code proposed in Algorithm (5), the number of exchanged packets is determined by the number of neighbors p of a node (line 4). In detail:

$$P_1(p) = O(p). \qquad (10.12)$$

During the "Orphan Recovery" step, according to the pseudo-code proposed in Algorithm (6), the number of exchanged packates is $O(p)$ for each attempt of attachment (line 3). In the worst case, p attempts are required in order to successfully attach an orphan (line 29). Consequently, the number of packets exchanged is:

$$P_2(p) = O(p^2). \tag{10.13}$$

Therefore, the overall number of packets required by the algorithm to build each path is given by:

$$\begin{aligned} P(N', p) &= N' \cdot [\alpha \cdot P_1(p) + (1 - \alpha) \cdot P_2(p)] & (10.14) \\ P(N', p) &= O(N' \cdot p^2) & (10.15) \end{aligned}$$

where, $N' = \frac{N}{k}$ is the average number of nodes of a path.

10.7.2 Algorithm II

Computational Load

During the "Approximate Hamiltonian Path Construction", according to the pseudo-code proposed in Algorithm (7), the dominant operation is the loop (lines: $7-13$) in which the candidate to be added is chosen. The related complexity is

$$C_1(p) = O(p) \tag{10.16}$$

where p is the max number of neighbors of each node.

During the "Path Refining process", according to the pseudo-code proposed in Algorithm (6), the dominant operation is the loop (lines: $7-22$) in which the orphan selects the path to be added to. The related complexity is equal to $O(p)$ where p is the max number of neighbors of each node. The worst case is achieved when the request of attachment for the orphan has to be sent p times. Thus, the complexity turns out to be

$$C_2(p) = O(p^2). \tag{10.17}$$

Therefore, the overall computational complexity is equal to:

$$\begin{aligned} C(N, p) &= N \cdot [\alpha \cdot C_1(p) + (1 - \alpha) \cdot C_2(p)] & (10.18) \\ C(N, p) &= O(N \cdot p^2) & (10.19) \end{aligned}$$

where, N is the number of nodes and α is the percentage of nodes covered at the first step, which can be retrieved by the statistical analysis.

Communication Load

During the "Approximate Hamiltonian Path Construction", according to the pseudo-code proposed in Algorithm (7), the number of exchanged packets is determined by the number of neighbors p of a node (line 4). In detail:

$$P_1(p) = O(p). \tag{10.20}$$

During the “Path Refining process”, according to the pseudo-code proposed in Algorithm (6), the number of exchanged packates is $O(p)$ for each attempt of attachment (line 3), where p is the number of neighbors of a node. In the worst case, p attempts are required in order to successfully attach an orphan (line 29). Consequently, the number of packets exchanged is:

$$P_2(p) = O(p^2). \tag{10.21}$$

Therefore, the overall number of packets required by the algorithm in order to build an approximate Hamiltonian path is given by:

$$\begin{aligned} P(N,p) &= N \cdot [\alpha \cdot P_1(p) + (1-\alpha) \cdot P_2(p)] && (10.22) \\ P(N,p) &= O(N \cdot p^2) && (10.23) \end{aligned}$$

where N is the number of nodes.

10.8 Consideration

The robot area coverage problem has been investigated by the robotics research community through the years. At the beginning, formulations involving a single robot were proposed, while more recently teams of robots have been taken into account.

In this work, a new formulation for the Multi-Robot Coverage problem is proposed. The novelty is the introduction of a sensor network, which cooperates with the robots in order to provide coordination to them. This is achieved taking advantage of the distributed nature of the sensor network. The sensor network is responsible for both the construction of the path and for guiding the robots.

This work focuses the attention on the distributed construction of paths. Two different algorithms have been provided. The first one speeds up the construction process exploiting a concurrent approach. While the second guarantees an underlying structure for the paths building a Hamiltonian path and then partitioning it.

A statistical analysis involving several configurations with different numbers of nodes and robots has been provided to validate the effectiveness of the proposed approaches. Three indexes of quality, namely completeness, fairness and robustness, have been exploited to investigate the performances.

According to the statistical results, both approaches proved to perform well. Moreover, they were shown to be complementary in the sense that the strength of one approach was the weakness of the other and vice-versa. In fact, the first algorithm, although it speeds up the the path construction process, turns out not to be optimal in case whether a robot leaves or joins the network. Conversely, the second approach, although it takes more time for the Hamiltonian path construction, provides *reactivity* in case a robot either joins or leaves the network.

Several challenges still remain for future work. From a theoretical standpoint, an analytical investigation for the bounds of performances should be considered. Moreover, a wider definition of robustness involving the failure of the sensor network, in terms of both communication and functioning, should be taken into account. Finally, a real implementation of these approaches in order to validate their effectiveness in a real context should be provided.

Chapter 11

Conclusions and Future Work

In the previous chapters the localization problem has been addressed in two different fields of interest, namely Mobile Robotics and Sensor Networks. For each field, the state of the art has been presented and different approaches have been proposed. The effectiveness of each approach has been validated by means of simulations and experimental results. Furthermore, an additional comparative analysis has been performed to highlight advantages and weaknesses of each single approach. In this chapter, conclusions are drawn and future work is discussed.

11.1 Conclusions

This research addressed the Localization Problem in two different fields of interest: Mobile Robotics and Sensor Networks. Indeed, the development of an integrated framework in which robotic components interact with sensor devices has caught the interest of the scientific community over the past decades. The availability of a reliable localization service is crucial in both fields: mobile robots must know their location to safely interact with their environment while sensor nodes are often required to be aware of their location (at least roughly) in order to properly supply services.

In the the robotic context, this research mainly focused on the global localization and the kidnapped robot problems. Three different approaches have been devised. The first approach presented in chapter 2 is an enhanced Monte Carlo Filter, namely the Clustered Evolutionary Monte Carlo Filter (CE-MCL), which was developed to overcome the classical Monte Carlo Filter drawbacks, for instance the depletion problem. This was achieved by introducing an evolutionary action along with a clusterization. The former was exploited to quickly find out local maxima while the latter to obtain an effective exploration of the environment. The second approach presented in chapter 3 is a Spatially Structured Genetic Algorithm (SSGA). The idea was to take advantage of the complex network theory for the spatial deployment of the population. In this way, two interesting advantages were achieved: a more effective exploration of the state space was achieved and the creation of evolutionary niches was improved. In particular, as niches represent regions in which particular solutions are preserved, a natural way to maintain the multi-hypothesis was obtained. The third approach presented in Chapter 4 is the Bacterial Colony Growth Algorithm (BCGA). This framework was inspired by the observation that some families of bacteria tend to form colonies when in presence of favorable conditions. Therefore, by modeling each bacterium as a hypothetical robot location and defining the favorable conditions with respect to the measurement match, a framework suitable for the global localization problem was obtained. The resulting framework features two different levels of modeling: a background level for maintaining the multi-hypothesis over time and a foreground level for selecting the best hypothesis at each time step. These approaches were compared in Chapter 5. The comparison was performed letting the robot ATRV-Jr manufactured by iRobot move within the Department of Computer Science and Automation of the University of “Roma Tre”. This environment was chosen for its particular structure. In fact, it is composed by two long corridors with several indentations leading to various symmetrical areas almost indistinguishable. A statistical analysis involving both mean and median values over 30 trials was performed. All the proposed approaches showed to effectively solve the global localization problem along with the kidnapped robot problem. More importantly, even if algorithms were sometimes temporarily deceived by the presence of several highly symmetrical areas, the exact location was always re-established taking advantage of the capability to carry on the multi-hypothesis over time.

In the Sensor Network context, this research addressed the Self-Localization problem. Two approaches have been proposed. The first approach presented in Chapter 7 is a Distributed Interlaced Kalman Filter. This approach was inspired by the multi-players dynamic game theory where the optimal solution is given by letting each player choose its strategy as optimal response to the strategy chosen by the other players. It consists of a set of reduced-order parallel Kalman Filters, each one running (on-board) of a node. Estimations are updated at each time-step by combining the latest observations with the latest estimates broadcasted within each neighborhood. The second approach presented in Chapter 8 is a Distributed Extended Information Filter. This approach based on the Information Theory is essentially a Kalman Filter expressed in terms of measures of information about the parameters (state) of interest (rather than direct state estimates and their associated covariance). Starting from a centralized formulation a distributed set of reduced-order filters was achieved by means of some simplifying assumptions. These approaches were compared in Chapter 9. The comparison was performed by exploiting a common sensor network deployment composed by 10 Micaz nodes produced by Crossbow. Experimental results highlighted comparable performances underlining the algebraic equivalence of the two formulations. Indeed, both approaches come from the Bayesian framework as they are derived from the "classical" Kalman Filter formulation. In particular, the Interlaced Kalman Filter performed slightly better in terms of estimation accuracy. This is due to the "interlacement" mechanisms which balances the approximation (introduced by the de-coupling operation) augmenting the noise covariance matrix of observations (by adding the cross-correlation terms). On the other hand, by moving the formulation to the Information domain, the Distributed Extended Information Filter showed an effective reduction of the computational complexity.

Finally, in Chapter 10 a possible integrated framework for the coverage problem was proposed. In this work the classical multi-robot scenario was extended by introducing a sensor network. The idea was to exploit the distributed nature of the sensor network to provide coordination to the robots. In particular, this work mainly focused on investigating how the dynamics of the coverage problem were modified by the introduction of the sensor network.

11.2 Future Work

The final objective is the construction of an integrated framework in which mobile robots take advantage of the availability of sensor devices to both extend their sensorial capability and distribute the computation over the network. This research represents a small step towards the construction of such a framework. Several other important problems need to be addressed. Among them, the clock synchronization problem is important to note. Indeed, the availability of a common clock is fundamental for applications, such as target tracking where data not only needs to be shared but also to be exactly located in the appropriate temporal frame.

In regard to the research proposed in this thesis several aspects need to be further investigated. In the robotic context, an experimental validation of the extensions devised for the multi-robot scenario should be faced. In the Sensor Network context, the proposed approaches should be experimentally validated in terms of scalability by exploiting networks with different orders of magnitude. For the integrated framework developed for the coverage problem, an analytical investigation of the performance boundaries should be considered. Moreover, a wider definition of robustness involving the failure of the sensor network, in terms of both communication and functioning, should be taken into account. Finally, a real implementation of this framework in order to validate its effectiveness in a real context should be provided.

Bibliography

[1] E. U. Acar and H. Choset. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *The International Journal of Robotics Research*, 21(4):345–366, April 2002. [cited at p. 109]

[2] Noa Agmon, Noam Hazon, and Gal A. Kaminka. Constructing spanning trees for efficient multi-robot coverage. In *ICRA 2006*, page In press, 2006. [cited at p. 108, 109, 113]

[3] H. Akashi and H. Kumamoto. State estimation for systems under measurements noise with markov dependent statistical property - an algorithm based on random sampling. In *6th Conf. IFAC*, 1975. [cited at p. 5]

[4] J. Allwright and D. Carpenter. A distributed implementation of simulated annealing for the traveling salesman problem. *Parallel Computing*, 10:335 – 338, 1989. [cited at p. 111]

[5] E.M. Arkin, S.P. Fekete, and J.S.B. Mitchell. The lawnmower problem. In *5th Canadian Conf. on Computational Geometry*, Waterloo, Ontario,, August 1993. [cited at p. 107]

[6] Esther M. Arkin, Sandor P. Fekete, and Joseph S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1-2):25–50, 2000. [cited at p. 108, 120]

[7] K.O. Arras, J.A. Castellanos, and R. Siegwart. Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints. In *Proceedings of the 2002 IEEE International Conference in Robotics and Automation*, volume 2, pages 1371–1377, 2002. [cited at p. 5]

[8] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 50(2):174–188, 2002. [cited at p. 6]

[9] Daniel Ashlock, Mark Smucker, and John Walker. Graph based genetic algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, pages 1362–1368, 1999. [cited at p. 27, 30]

[10] D. Austin and P. Jensfelt. Using multiple gaussian hypotheses to represent probability distributions for mobile robot localization. In *In Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, CA., 2000. [cited at p. 5]

[11] Jeremy Avigad and Kevin Donnelly. Formalizing o notation in isabelle/hol. In David Basin and Michaël Rusinowitch, editors, *Automated Reasoning: Second International Joint Conference, IJCAR 2004*. SpringerVerlag, 2004. [cited at p. 68, 100]

[12] P. Bahl and V. N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM*, volume 2, pages 775–784, 2000. [cited at p. 72]

[13] Jrgen Bang-Jensen and Gregory Gutin, editors. *Digraphs: Theory, Algorithms and Applications*. Springer-Verlag, London Springer Monographs in Mathematics, 2000. [cited at p. 111]

[14] Y. Bar-Shalom and T.E. Fortman. *Tracking and Data Association*. Academic Press, 1988. [cited at p. 89]

[15] Albert-Laszlo Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999. [cited at p. 26]

[16] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328 – 342, June 1995. [cited at p. 3, 4]

[17] M. Batalin and G. Sukhatme. Spreading out: A local approach to multi-robot coverage. In *6th International Symposium on Distributed Autonomous Robotic Systems*, pages 373–382, 2002. [cited at p. 108, 110]

[18] M. Batalin and G. S. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems Journal*, 26(2):181 – 196, 2004. Special Issue on Wireless Sensor Networks. [cited at p. 112]

[19] M. Batalin and G. S. Sukhatme. The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment. *IEEE Transactions on Robotics*, 23(4):661 – 675, 2007. [cited at p. 112]

[20] E.T. Baumgartner and S.B. Skaar. An autonomous vision-based mobile robot. *IEEE Transactions on Automatic Control*, 39(3):493–502, March 1994. [cited at p. 4]

[21] John Beasley. Route first - cluster second methods for vehicle routing. *Omega*, 11(4):403 – 408, 1983. [cited at p. 116]

[22] Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega: The International Journal of Management Science*, 34(3):209–129, 2006. [cited at p. 111]

[23] P. Biswas, T. C. Liang, K. C Toh ., Y. Ye, and T. C. Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *Automation Science and Engineering, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 3(4):360–371, 2006. [cited at p. 73]

[24] F. Bock. An algorithm for solving traveling-salesman and related network optimization problems. Unpublished manuscript associated with talk presented at the 14th ORSA National Meeting, 1965. [cited at p. 111]

[25] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989. [cited at p. 5]

[26] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In *Proceedings of the 1990 IEEE International Conference in Robotics and Automation*, volume 1, pages 572–577, May 1990. [cited at p. 5]

[27] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991. [cited at p. 5]

[28] Ingwer Borg and Patrick Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer-Verlag New York, Inc., first edition edition, 1996. (Springer Series in Statistics). [cited at p. 73]

[29] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001. [cited at p. 72]

[30] Jehoshua Bruck, Jie Gao, and Anxiao (Andrew) Jiang. Localization and routing in sensor networks by local angle information. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 181–192, New York, NY, USA, 2005. ACM Press. [cited at p. 72]

[31] R. Bucher and D. Misra. A synthesizable vhdl model of the exact solution for three-dimensional hyperbolic positioning system. *VLSI Design*, 15(2):507 – 520, 2002. [cited at p. 149]

[32] Marco Budinich. Neural networks for the travelling salesman problem. *J. Artif. Neural Netw.*, 2(4):431–435, 1995. [cited at p. 111]

[33] J.M. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S. Thrun. The mobile robot RHINO. *AI Magazine*, 16(2):31–38, 1995. [cited at p. 5]

[34] Nirupama Bulusu, Vladimir Bychkovskiy, Deborah Estrin, and John Heidemann. Scalable, ad hoc deployable, rf-based localization. In *Proceedings of the Grace Hopper Celebration of Women in Computing*, Vancouver, British Columbia, Canada, October 2002. [cited at p. 74]

[35] Nirupama Bulusu, Deborah Estrin, Lewis Girod, and John Heidemann. Scalable coordination for wireless sensor networks: Self-configuring localization systems. In *Proceedings of the Sixth International Symposium on Communication Theory and Applications (ISCTA '01)*. University of California, Los Angeles, July 2001. [cited at p. 73]

[36] W. Burgard, A. Derr, D. Fox, and A. Cremers. Integrating global position estimation and position tracking for mobile robots: the dynamic markov localization approach. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '98),*, pages 730–735, Victoria, BC, Canada, October 1998. [cited at p. 5]

[37] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. of the Fourteenth National Conference on Artificial Intelligence*, volume 2, pages 896–901, 1996. [cited at p. 5]

[38] K. Burrage and P. M. Burrage. *Numerical methods for stochastic differential equations with application*. SIAM, 2003. [cited at p. 39]

[39] Zack Butler, Alfred Rizzi, and Ralph Hollis. Contact sensor-based coverage of rectilinear environments. In *IEEE Int'l Symposium on Intelligent Control*, pages 266 – 271, September 1999. [cited at p. 110]

[40] Zack Butler, Alfred Rizzi, and Ralph Hollis. Distributed coverage of rectilinear environments. In *Proc. of the Workshop on the Algorithmic Foundations of Robotics*. A. K. Peters, January 2001. [cited at p. 108, 110]

[41] Y. Uny Cao, Alex S. Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Auton. Robots*, 4(1):7–27, 1997. [cited at p. 7]

[42] Alberto Cerpa, Jeremy Elson, Michael Hamilton, Jerry Zhao, Deborah Estrin, and Lewis Girod. Habitat monitoring: application driver for wireless communications technology. In *SIGCOMM LA '01: Workshop on Data communication in Latin America and the Caribbean*, pages 20–41, New York, NY, USA, 2001. ACM. [cited at p. 71]

[43] Haowen Chan, Mark Luk, and Adrian Perrig. Using clustering information for sensor network localization. In *The International Conference on Distributed Computing in Sensor Systems (DCOSS 2005)*, 2005. [cited at p. 72, 74]

[44] Luo Chaomin and S.X. Yang. A real-time cooperative sweeping strategy for multiple cleaning robots. In *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, pages 660 – 665, 2002. [cited at p. 108, 110]

[45] B.H. Cheng, R.E. Hudson, F. Lorenzelli, L. Vandenberghe, and K. Yao. Distributed gauss-newton method for node localization in wireless sensor networks. In *6th Workshop on Signal Processing Advances in Wireless Communications, 2005 IEEE*, pages 915–919, June 2005. [cited at p. 72]

[46] H. Choset. Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113 – 126, October 2001. [cited at p. 108]

[47] H. Choset and P. Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *International Conference on Field and Service Robotics*, Canberra, Australia, 1997. [cited at p. 109]

[48] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, CMU, 1976. [cited at p. 111]

[49] G. Clarke and G.W. Wright. Scheduling of vehicles from a central depot to number of delivery points. *Operations Research*, 12:568–581, 1964. [cited at p. 111]

[50] J. Colegrave and A. Branch. A case study of autonomous household vacuum cleaner. In AIAA/NASA CIRFFSS, 1994. [cited at p. 107]

[51] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251–280, 1990. [cited at p. 79]

[52] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London: John Murray, 1859. [cited at p. 144]

[53] M. Diaby. The traveling salesman problem: A linear programming formulation. *WSEAS Transactions on Mathematics*, 6(6), June 2007. [cited at p. 111]

[54] L. Doherty, K.S.J. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *NFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1655 – 1663, 2001. [cited at p. 72, 73]

[55] Jim Doll and David L. Freeman. Monte carlo methods in chemistry. *IEEE Comput. Sci. Eng.*, 1(1):22–32, 1994. [cited at p. 6]

[56] Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997. [cited at p. 111]

[57] A. Doucet. *Monte carlo methods for bayesian estimation of hidden markov models. applications to radiation signals*. PhD thesis, Univ. Paris-Sud, Orsay, France, 1997. [cited at p. 6]

[58] Arnaud Doucet. On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR. 310, Cambridge University Department of Engineering, Signal Processing Group, 1998. [cited at p. 6]

[59] Arnaud Doucet, Nando De Freitas, and Neil Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer-Verlag New York, Inc., 2001. [cited at p. 5]

[60] H Durrant-Whyte. Uncertain geometry in robotics. In *Proceedings of the 1987 IEEE International Conference in Robotics and Automation*, volume 4, pages 851 –856, Marchl 1987. [cited at p. 4]

[61] Hugh F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1987. [cited at p. 4]

[62] T. Eren, D. Goldenberg, W. Whitley, Y. Yang, A. Morse, B. Anderson, and P. Belheumer. Rigidity, computation, and randomization of network localization. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2673 – 2684, March 2004. [cited at p. 72]

[63] Mehdi Essoloh, Cedric Richard, and Hichem Snoussi. Anchor-based distributed localization in wireless sensor networks. In *Workshop on Statistical Signal Processing, 2007. SSP '07. IEEE/SP 14th*, pages 393 – 397, August 2007. [cited at p. 73]

[64] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press. [cited at p. 14, 15]

[65] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344, 2000. [cited at p. 8]

[66] Dieter Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, Diss, University of Bonn, Germany, 1998. [cited at p. 10, 11]

[67] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99).*, July 1999. [cited at p. 5, 6]

[68] Yoav Gabriely and Elon Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):77–98, 2001. [cited at p. 108, 109]

[69] D. W. Gage. Randomized search strategies with imperfect sensors. In W. H. Chun and W. J. Wolfe, editors, *Proc. SPIE Vol. 2058, p. 270-279, Mobile Robots VIII, Wendell H. Chun; William J. Wolfe; Eds.*, volume 2058 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 270–279, February 1994. [cited at p. 107]

[70] M. Giacobini, M. Tomassini, and A. Tettamanzi. Takeover time curves in random and small-world structured populations. In *Proc. of the 2005 conference on Genetic and evolutionary computation*, pages 1333 – 1340, 2005. [cited at p. 31]

[71] L. Glielmo, R. Setola, and F. Vasca. An interlaced extended kalman filter. *IEEE Transactions on Automatic Control*, 44(8):1546 – 1549, August 1999. [cited at p. 76]

[72] Luigi Glielmo, Roberto Setola, and Francesco Vasca. nterlaced extended kalman filter as deterministic nonlinear observer. In *Proc. of European Control Conference ECC'99*, Karlsruhe, Germany, 1999. [cited at p. 78]

[73] P. Goel, S.I. Roumeliotis, and G.S. Sukhatme. Robust localization using relative and absolute position estimates. In *Proceedings. 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999. IROS '99.*, volume 2, pages 1134–1140, Marchl 1999. [cited at p. 4]

[74] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. [cited at p. 144]

[75] S. Grime, H.F. Durrant-Whyte, and P. Ho. Communication in decentralized sensing. Technical report, Oxford University Robotics Research Group, 1991. [cited at p. 87]

[76] J. Handschin. Monte carlo techniques for prediction and filtering on non-linear stochastic processes. *Automatica*, 6:555–563, 1970. [cited at p. 5]

[77] J. Handschin and D. Mayne. Monte carlo techniques to estimate the conditional expectation in multi-stage non linear filtering. *Int. J. Cont*, 9(5):547–559, 1969. [cited at p. 5]

[78] J. Handshin. Monte carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, 6:555 – 563, 1970. [cited at p. 12]

[79] Noam Hazon and Gal A. Kaminka. Redundancy, efficiency, and robustness in multi-robot coverage. In *ICRA 2005*, 2005. [cited at p. 108, 109, 113]

[80] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-free localization schemes for large scale sensor networks. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 81–95, New York, NY, USA, 2003. ACM Press. [cited at p. 72, 74]

[81] John H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975. [cited at p. 144]

[82] A. Howard, M. J. Mataric, and G. S. Sukhatme. *Experimental Robotics VIII*, chapter Localization for Mobile Robot Teams: A Distributed MLE Approach, pages 146–155. Springer Berlin / Heidelberg, 2003. [cited at p. 9]

[83] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. [cited at p. 6]

[84] Bill Jackson and Tibor Jordán. Connected rigidity matroids and unique realizations of graphs. *J. Comb. Theory Ser. B*, 94(1):1–29, 2005. [cited at p. 72]

[85] P. Jensfelt and S. Kristensen. Active global localization for a mobile robot using multiplehypothesis tracking. *IEEE Transaction on Robotics and Automation (ICRA'01)*, 17(5):748 – 760, October 2001. [cited at p. 5]

[86] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions ASME Journal of Basic Engineering*, 82:35 – 44, 1960. [cited at p. 3]

[87] L. L. Karg and G. L. Thompson. A heuristic approach to traveling salesman problems. *Management Sci.*, 10:225 – 248, 1964. [cited at p. 111]

[88] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 254–263, New York, NY, USA, 2007. ACM. [cited at p. 71]

[89] Y. J. Kim, R. Govindan, B. Karp, and S. Shenker. Lazy Cross-Link Removal for Geographic Routing. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (Sensys)*, Boulder, Colorado, USA, November 2006. [cited at p. 121]

[90] Y.J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 16–16, Berkeley, CA, USA, 2005. USENIX Association. [cited at p. 121]

[91] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983. [cited at p. 111]

[92] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, March 1996. [cited at p. 6]

[93] C. S. Kong, N. A. Peng, and I. Rekleitis. Distributed coverage with multi-robot system. In *IEEE International Conference on Robotics and Automation. ICRA-2006.*, pages 2423–2429, May 2006. [cited at p. 108, 109]

[94] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72, New York, NY, USA, 2003. ACM Press. [cited at p. 72]

[95] D. Kurabayashi, J. Ota, and E. Yoshida. An algorithm of dividing a work area to multiple mobile robots. In *IROS '95: Proceedings of the International Conference on Intelligent Robots and Systems-Volume 2*, page 2286, Washington, DC, USA, 1995. IEEE Computer Society. [cited at p. 108, 110]

[96] R. Kurazume, S. Nagata, and S. Hirose. Cooperative positioning with multiple robots. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1250–1257, September, 8–13 1994. [cited at p. 8]

[97] C. Kwok, D. Fox, and M. Meila. Real-time particle filters. *Proceedings of the IEEE*, 92(4):469 – 484, March 2004. [cited at p. 5, 6]

[98] N.M. Kwok, G. Fang, and W. Zhou. Evolutionary particle filter: Re-sampling from the genetic algorithm perspectve. In *EEE/RSJ International Conference on IROS*, pages 2935 – 2940, August 2005. [cited at p. 6, 7]

[99] Gilbert Laporte. Generalized subtour elimination constraints and connectivity constraints. *The Journal of the Operational Research Society*, 37(5):509–514, May 1986. [cited at p. 124]

[100] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376 – 382, June 1991. [cited at p. 4]

[101] Victor Lesser, Michael Atighetchi, Brett Benyo, Bryan Horling, Anita Raja, Regis Vincent, Thomas Wagner, Xuan Ping, and Shelley XQ Zhang. The intelligent home testbed. *Proceedings of the Autonomy Control Software Workshop (Autonomous Agent Workshop)*, January 1999. [cited at p. 72]

[102] Chong Liu, Kui Wu, and Tian He. Sensor localization with ring overlapping based on comparison of received signal strength indicator. In *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 516 – 518, October 2004. [cited at p. 72]

[103] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998. [cited at p. 6]

[104] T. Malthus. *An essay on the principle of population*. London, printed for J. Johnson, in St. Paul's Church-Yard, 1798. [cited at p. 39]

[105] A. Martinelli, F. Pont, and R. Siegwart. Multi-robot localization using relative observation. In *Proc. of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 18–22 2005. [cited at p. 8]

[106] A. Martinelli and R. Siegwart. Observability analysis for mobile robot localization. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robot and System*, pages 1264–1269, Edmonton, Canada, August 2–6 2005. [cited at p. 8]

[107] Peter S. Maybeck. *Stochastic models estimation and control*, volume 1. Academic Press, 1979. [cited at p. 3, 89]

[108] Vipin Mehta and Magda El Zarki. A bluetooth based sensor network for civil infrastructure health monitoring. *Wirel. Netw.*, 10(4):401–412, 2004. [cited at p. 71]

[109] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, 1960. [cited at p. 111]

[110] Adam Milstein, Javier Nicolás Sánchez, and Evan Tang Williamson. Robust global localization using clustered particle filtering. In *Eighteenth national conference on Artificial intelligence*, pages 581–586, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. [cited at p. 6]

[111] Tao Wei Min and How Khee Yin. A decentralized approach for cooperative sweeping by multiple mobile robots. In *International Conference on Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ*, pages 380 –385, 1998. [cited at p. 108, 110]

[112] David Moore, John Leonard, Daniela Rus, and Seth Teller. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, New York, NY, USA, 2004. ACM Press. [cited at p. 72, 73]

[113] Hans Moravec. Sensor fusion in certainty grids for mobile robots. *AI Mag.*, 9(2):61–74, 1988. [cited at p. 5]

[114] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *In Proc. IEEE Int. Conf. Robotics and Automation*, pages 116 – 121, 1985. [cited at p. 5]

[115] A.I. Mourikis and S.I. Roumeliotis. Performance analysis of multirobot cooperative localization. *IEEE Trans. on Robotics*, 22(4):666–681, August 2006. [cited at p. 8]

[116] H. Muhlenbein, M. Georges-Schleuter, and O. Kramer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7, 1988. [cited at p. 111]

[117] Arthur G. O. Mutambara. *Decentralized Estimation and Control for Multisensor Systems.* CRC Press, Inc., Boca Raton, FL, USA, 1998. [cited at p. 88, 90]

[118] John Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950. [cited at p. 76]

[119] D. Niculescu and Badri Nath. Ad hoc positioning system (aps) using aoa. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE (INFOCOM 2003)*, volume 3, pages 1734 – 1743, April 2003. [cited at p. 72]

[120] Erdos P. and Renyi A. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959. [cited at p. 25]

[121] S. Panzieri, F. Pascucci, and G. Ulivi. An outdoor navigation system using gps and inertial platform. *IEEE/ASME Trans. on Mechatronics*, 7(2):134–142, 2002. [cited at p. 3, 4]

[122] L. E. Parker. Current state of the art in distributed robot systems. In *Proceedings of Distributed Autonomous Robotic Systems 4*, pages 3–12, 2000. [cited at p. 8]

[123] Shyamal Patel, Konrad Lorincz, Richard Hughes, Nancy Huggins, John H. Growdon, Matt Welsh, and Paolo Bonato. Analysis of feature space for monitoring persons with parkinson's disease with application to a wireless wearable sensor system. In *In Proceedings of the 29th IEEE EMBS Annual International Conference*, Lyon, France, August 2007. [cited at p. 72]

[124] N. Patwari, A. III, M. Perkins, N. Correal, and R. O'Dea. Relative location estimation in wireless sensor networks. *IEEE Transaction on Signal Processing*, 51(8):2137–2148, August 2003. [cited at p. 72]

[125] A. L. Pearce, P. E. Rybski, S. A. Stoeter, and N. Papanikolopoulos. Dispersion behaviors for a team of multiple miniature robots. In *International Conference on Robotics and Automation, ICRA-2003*, pages 1158 – 1163, Taipei, Taiwan, September 2003. [cited at p. 107]

[126] Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, and Seth Teller. Poster abstract: anchor-free distributed localization in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 340–341, New York, NY, USA, 2003. ACM. [cited at p. 73, 74]

[127] I. Rekleitis, G. Dudeck, and E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1340–1345, 1997. [cited at p. 8]

[128] I. Rekleitis, V. Lee-Shue, Ai P. New, and H. Choset. Limited communication, multi-robot team based coverage. In *International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE*, volume 4, pages 3462–3468, 2004. [cited at p. 108]

[129] I. M. Rekleitis, G. Dudek, and E. E. Milios. Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robot and System*, pages 2690–2696, Lausanne, Switzerland, September 30–October 4 2002. [cited at p. 8]

[130] I. M. Rekleitis, G. Dudek, and E. E. Milios. Probabilistic cooperative localization and mapping in practice. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1907–1912, Taipei, Taiwan, September 14–19 2003. [cited at p. 8]

[131] Jonathan Rogge and Dirk Aeyels. A novel strategy for exploration with multiple robots. In *Proceedings of the 4th International Conference On Informatics in Control, Automation and Robotics*, Angers, France, 2007. [cited at p. 108, 109]

[132] D.J Rosenkrantz, R.E Stearns, and P.M. Lewis I. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563 – 581, 1977. [cited at p. 111]

[133] S. I. Roumeliotis and G. A. Bekey. Distribuited multirobot localization. *IEEE Trans. on Robotics and Automation*, 18(5):781–795, October 2002. [cited at p. 8, 78]

[134] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Smoother based 3-d attitude estimation for mobile robot localization. In *Proceedings of the 1999 IEEE International Conference in Robotics and Automation*, volume 3, pages 1979–1986, May 1999. [cited at p. 3, 4]

[135] Stergios I. Roumeliotis and Ioannis M. Rekleitis. Propagation of uncertainty in cooperative multirobot localization: Analysis and experimental results. *Auton. Robots*, 17(1):41–54, 2004. [cited at p. 9]

[136] Chris Savarese, Jan M. Rabaey, and Koen Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 317–327, Berkeley, CA, USA, 2002. USENIX Association. [cited at p. 73]

[137] S. Schnell and T. E. Turner. Reaction kinetics in intracellular environments with macromolecular crowding: Simulations and rate laws. In *Prog Biophys Mol Bio*, pages 235 – 260, 2004. [cited at p. 39]

[138] G. A. Sena, D. Megherbi, and G. Isern. Implementation of a parallel genetic algorithm on a cluster of workstations: traveling salesman problem, a case study. *Future Gener. Comput. Syst.*, 17(4):477–488, 2001. [cited at p. 111]

[139] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization from mere connectivity. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 201–212, New York, NY, USA, 2003. ACM Press. [cited at p. 72, 73]

[140] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(4):623–656, 1948. [cited at p. 18]

[141] Xingfa Shen, Zhi Wang, Peng Jiang, Ruizhong Lin, and Youxian Sun. *Advances in Intelligent Computing*, chapter Connectivity and RSSI Based Localization Scheme for Wireless Sensor Networks, pages 578 – 587. Springer Berlin / Heidelberg, 2005. [cited at p. 72]

[142] Victor Shnayder, Bor rong Chen, Konrad Lorincz, Thaddeus R. F. Fulford Jones, and Matt Welsh. Sensor networks for medical care. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 314–314, New York, NY, USA, 2005. ACM. [cited at p. 72]

[143] Sam Ge Shuzhi and C. Fua. Complete multi-robot coverage of unknown environments with minimum repeated coverage. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 715 – 720, 2005. [cited at p. 108, 109]

[144] Mani B. Srivastava, Richard R. Muntz, and Miodrag Potkonjak. Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving enviroments. In *Mobile Computing and Networking*, pages 132–138, 2001. [cited at p. 72]

[145] I. Stojmenovic. Position-based routing in ad hoc networks. *Communications Magazine, IEEE*, 40(7):128–134, 2002. [cited at p. 72]

[146] Marco Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. [cited at p. 27]

[147] Stefan Tschöke, Reinhard Lüling, and Burkhard Monien. Solving the traveling salesman problem with a distributed branch-and-bound algorithm on a 1024 processor network. In *IPPS '95: Proceedings of the 9th International Symposium on Parallel Processing*, pages 182–189, Washington, DC, USA, 1995. IEEE Computer Society. [cited at p. 111]

[148] P. F. Verhulst. Recherches matematiques sur la loi d'accroissement de la population. *Noveaux Memories de l'Academie Royale des Sciences et Belles-Lettres de Bruxelles*, 18(1):1 – 45, 1845. [cited at p. 39]

[149] T. Volgenant and R. Jonker. A branch and bound algorithm for the symmetric traveling salesman problem based on the 1-tree relaxation. *European Journal of Operations Research*, 9:83 – 89, 1982. [cited at p. 111]

[150] V. Volterra. Variations and fluctuations of the number of individuals in animal species living... *Animal Ecology*, 1, 1931. [cited at p. 39, 40]

[151] I.A. Wagner, Lindenbaum M, and A.M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *Robotics and Automation, IEEE Transactions on*, 15(5):918–933, October 1999. [cited at p. 108, 110]

[152] X. Wang, R. Chen, and J. Liu. Monte carlo bayesian signal processing for wireless communications. *Journal of VLSI Signal Processing*, 30(29):89 – 105, 2002. [cited at p. 6]

[153] Xiao Fan Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine, IEEE*, 3(1):6–20, 2003. [cited at p. 27]

[154] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998. [cited at p. 26]

[155] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006. [cited at p. 71, 72]

[156] Kamin Whitehouse, Chris Karlof, Alec Woo, Fred Jiang, and David Culler. The effects of ranging noise on multihop localization: an empirical study. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, pages 73–80, Piscataway, NJ, USA, 2005. IEEE Press. [cited at p. 72]

[157] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80 – 83, December 1945. [cited at p. 46, 58]

[158] Kiran Yedavalli, Bhaskar Krishnamachari, Sharmila Ravula, and Bhaskar Srinivasan. Ecolocation: A sequence based technique for rf-only localization in wireless sensor networks. In *The Fourth International Conference on Information Processing in Sensor Networks (IPSN '05)*, pages 285 – 292, April 2005. [cited at p. 72, 74]

[159] A. Youssef, A. Ashok, and M. Younis. Accurate anchor-free node localization in wireless sensor networks. In *24th IEEE International Performance, Computing, and Communications Conference, 2005. IPCCC 2005.*, pages 465 – 470, April 2005. [cited at p. 73, 75]

[160] V. Zaritskii, V. Svetnik, and L. Shimelevich. Monte carlo techinique in problems of optimal data processing. *Auto. Remo. Cont.*, 12:95–103, 1975. [cited at p. 6]

[161] X. Zheng, S. Jain, S. Koenig, and D. Kempe. Multi-robot forest coverage. In *IROS 2005*, 2005. [cited at p. 108, 110]

[162] J. Zheng-Wei and G. Yuan-Tao. Novel adaptive particle filters in robot localization. *ACTA Automatica Sinica*, 31(6):833–838, November 2005. [cited at p. 5, 6]

Appendices

Appendix A

Robot, Sensors and Environment Modelling

A.1 Robot kinematics: The unicycle model

The robot pose can be uniquely determined in an environment by means of the robot position (x, y) and orientation (ϕ). Here, the unicycle model has been adopted as the kinematic model for the robot. Such a model is described as follows:

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}, n_{k-1}) \\ &= x_{k-1} + \begin{bmatrix} \cos\tilde{\phi}_{k-1} & 0 \\ \sin\tilde{\phi}_{k-1} & 0 \\ 0 & 1 \end{bmatrix} u_{k-1} + n_{k-1} \end{aligned} \tag{A.1}$$

where, $x_k = [r_x, r_y, r_\phi]$ is the robot pose at time-step k (*state*), u_{k-1} is the input at time $k-1$ and n_{k-1} is a white zero mean noise at the same time-step. In particular, the system input is $u_k = (\delta s_k, \delta\theta_k)$, where δs_k is the vehicle displacement and $\delta\theta_k$ is the rotation during the sample time interval δt_k, both measured by proprioceptive sensors.

A.2 Sensors and Environment Modelling

The robot has been equipped with a set of laser rangefinders arranged in a 360° pattern. The related observation model, taking into account the fact that the environment has been described through a set $\mathcal{M}$ of segments, is:

$$\begin{aligned} z_{j,k} &= h(x_k, \mathcal{M}) \\ &= \frac{|a_r l_j^x + b_r l_j^y + c_r|}{|a_r \cos\theta_j + b_r \sin\theta_j|} \end{aligned} \tag{A.2}$$

where (a_r, b_r, c_r) are the coefficients of the r-th segment and (l_j^x, l_j^y, θ_j) is the configuration of the laser beam detecting the segment in question.

Appendix B

Genetic Algorithms

Genetic Algorithms (GAs) are a class of search methods and computational models inspired by Darwin's *Theory of Evolution*. These algorithms, initially investigated in [81], use a population to explore the search space, by means of probabilistic transition operators like crossover and mutation, in order to find out the element (*chromosome*) that best fits a given objective function (*fitness function*). This approach reflects a possible mathematical modeling of the *nature's behavior* in which the high adaptability of each creature in its environment is the result of a long evolutionary process, based on natural selection, mutation, sexual and asexual reproduction [52]. GAs have been applied in several research areas to solve optimization problems where the presence of non-differentiable or non-continuous objective functions makes other methodologies almost useless.

B.1 A Simple Genetic Algorithm (SGA)

A simple genetic algorithm, as it is referred to in [74], usually provides three steps: initialization, selection and reproduction. The pseudocode in Algorithm 8 shows a possible implementation schema for an SGA, where the roulette wheel is exploited for selection and crossover and mutation are used for reproduction.

B.1.1 Initialization

Initialization generates an initial population whose elements are encoded by means of a fixed length string known in literature as *genotype*, or alternatively *chromosome*. Several strategies have been proposed for the initialization; a classical one is to randomly draw the population. Afterward, the fitness function has to be evaluated for each element of the population. The identification of a suitable objective function, able to give a measure of the goodness of an element, is usually problem-dependent.

Algorithm 8: A Simple Genetic Algorithm Schema

```
Data: Fitness function $f(\cdot)$
Result: $p_k^*$
/* Initialisation                                        */
Set $k = 0$; Create $P_k = \{p_{1,k}, \dots, p_{n,k}\}$
while $StopCondition(\tilde{p}_{j,k})$ do
    /* Roulette Wheel Selection                          */
    for $i$=1 to $n$ do
        $x = random(0, 1)$;
        $j = 1$;
        while $j < n \,\&\, x < \frac{\sum_{l=1}^{j} f(p_{l,k})}{\sum_{l=1}^{n} f(p_{l,k})}$ do
            $j = j + 1$;
        end
        $\tilde{p}_i = p_{j,k}$;
    end
    /* Crossover Reproduction                            */
    for $i$=1 to $n$-1 do
        $p_{i,k+1} = Crossover(\tilde{p}_i, \tilde{p}_{i+1})$
    end
    /* Mutation Reproduction                             */
    for $i$=1 to $n$ do
        $p_{i,k+1} = Mutation(\tilde{p}_i)$
    end
    $p_{k+1}^* = max_{f(\cdot)}\{\{p_{1,k+1}, \dots, p_{n,k+1}\}\}$;
    $k = k + 1$;
end
```

B.1.2 Selection

Selection draws an intermediate population where the recombination has to be applied by preferring individuals with higher fitness over low-fitted ones. Although it can be a deterministic operator, often some random components are considered. A very popular probabilistic schema is the roulette wheel selection. The idea is to relate the probability to choose a certain individual to its fitness. From a probabilistic standpoint, it can be expressed as a random experiment, where the probability of the individual i to be selected at time k is given by:

$$P(b_{i,k}) = \frac{f_{b_{i,k}}}{\sum_{j=1}^{m} b_{j,k}} \tag{B.1}$$

This schema can be graphically described as a “tricked” fortune wheel where the slots are not equally wide and therefore different outcomes can occur with different probabilities. Note that, this formula can be applied only if the fitness value are positive. Otherwise, a proper non-decreasing transformation is required.

B.1.3 Reproduction: Crossover and Mutation

Reproduction acts on the intermediate population in order to generate a new one. Probabilistic transition operators crossover and mutation, are usually applied.

Crossover

Crossover mimics the sexual reproduction as it appears in the real world: the genetic material of two parents is mixed when the gametes of the parents merge. The interesting property of this operator, along with the capability to introduce new genetic material and maintain the genetic diversity, is the capability to produce in average better generations. Several implementations of this operator can be provided such as the one-point crossover or the n-point crossover. The one-point crossover which creates a new element swapping a portion of parents chromosomes with respect to a cutting point is the simplest implementation.

Mutation

Mutation mimics the biological mutation of the genetic material which may occur to an organism for several reasons such as an exposure to ultraviolet or ionizing radiation or deliberately for instance as a consequence of an environmental adaptation process. From an algorithmic standpoint, this operator is fundamental to prevent the population from stagnating at any local optima. Like for the crossover operator, also in this case several implementation can be provided. A classical implementation involves a probability that an arbitrary bit (or a sequence of bits) may be changed from their initial value(s).

Appendix C

Sensor Network Scenario Modeling

The deployment of static sensor network consisting of a group of Ω nodes on a planar environment is considered. A typical sensor network node's hardware consists of a microprocessor with reduced computational capability, a radio component, several sensor devices, a minimal data storage unit and a battery with limited life. In addition, a few nodes are supposed to be equipped with an absolute position system device so that localization with respect to a global reference frame can be obtained for the whole network. Communication among nodes is possible only if they are within the coverage area of each other.

In this context, the state of the node i at time k is described by its location with respect to a global reference frame as follows:

$$x_k^{(i)} = [\,p_{x,k}^{(i)}\; p_{y,k}^{(i)}\,] \tag{C.1}$$

Therefore, the state of the whole system is the vector obtained by collecting the locations of all nodes:

$$\mathbf{x}_k = \begin{bmatrix} x_k^{(1)} \\ x_k^{(2)} \\ \vdots \\ x_k^{(\Omega)} \end{bmatrix} \tag{C.2}$$

C.1 System model

Since the network is assumed to be static, the model of the i-th node is simply given by:

$$x_k^{(i)} = F_k^{(i)}\, x_{k-1}^{(i)} + w_k^{(i)} \tag{C.3}$$

where $F_k^{(i)} = I_2 \in \mathcal{R}^{2\times 2}$ is the identity transition matrix and $w_k^{(i)} \sim \mathcal{N}(0, Q_k^{(i)}) \in \mathcal{R}^2$ is a zero mean white noise vector with covariance matrix $Q_k^{(i)}$. Note that, the system is naturally fully decoupled as the state transition of a node does not depend upon other nodes.

C.2 Observation model

Nodes are equipped with several sensor devices. In particular, for all these nodes within the coverage area of each other, a way to measure inter-node distances is assumed to be available. The related observation model can be obtained considering the Euclidian distance as follows:

$$\begin{aligned} z_k^{(i,j)} &= h(x_k^{(i)}, x_k^{(j)}) + v_k^{(i)} && \text{(C.4)} \\ &= \parallel x_k^{(i)} - x_k^{(j)} \parallel + v_k^{(i)} && \text{(C.5)} \\ &= \sqrt{(p_{x,k}^{(i)} - p_{x,k}^{(j)})^2 + (p_{y,k}^{(i)} - p_{y,k}^{(j)})^2} + v_k^{(i)} && \text{(C.6)} \end{aligned}$$

where $\parallel \cdot \parallel$ is the Euclidean distance and $v_k^{(i)} \sim \mathcal{N}(0, R_k^{(i)}) \in \mathcal{R}^2$ is a zero mean white noise vector with covariance matrix $R_k^{(i)}$.

Appendix D

Ranging Technique for MICAz

The Time Difference of Arrival (TDoA) is a technique commonly used in civil and military surveillance applications to accurately locate targets by determining the difference between the time of arrival of two pulses characterized by a different propagation velocity [31].

D.1 Time Difference of Arrival for MICAz

The MICAz (MPR2400) platform is a generation of Motes from Crossbow Technology. The MPR2400 (2400 MHz to 2483.5 MHz band) uses the Chipcon CC2420, IEEE 802.15.4 compliant, ZigBee ready radio frequency transceiver integrated with an Atmega128L micro-controller. It provides also a flash serial memory, as well as a 51 pin I/O connector that allows several sensor and data acquiring boards to be connected to it. In particular, two different boards – the MTS300 and the MTS310 – both provide a sounder as well as a microphone. The sounder is a simple 4 kHz fixed frequency piezoelectric resonator, while the microphone can be used either for acoustic ranging or for general acoustic recording and measurement. Therefore, the RF and acoustic (sounder) signals can be exploited for the TDoA.

D.2 TDoA Analysis

The proposed TDoA for MICAz platforms has been thoroughly investigated in order to determine the achievable performance. A significant amount of inter-node distances (more than 200 measurements) were collected and a statistical analysis was performed. A precision of $3 \sim 8$ cm with a standard deviation of $8 \sim 14$ cm was experienced considering distances ranging from 20 cm to 2.5 m.

In addition, experiments were carried out to verify whether the mutual orientation of nodes might influence the measured distance. For such a reason, two nodes were arranged on the floor at the distance of 54 cm from each other. Such a distance was manually measured from the sounder of the emitter to the microphone of the receiver. Successively, data was collected considering different orientations of nodes, in order to simulate a realistic random deployment on the ground. Table D.1 shows the statistic results using again more than 200 measurements for each configuration. According to this analysis differential mutual orientations do not significantly influence the measure of distances. However, as mentioned above, data presents a bias as well as a considerable standard deviation that makes their use challenging.

Exp.	mean value	std dev	node 1 orientation	node 2 orientation
1	0.5781	0.1229	$\pi/2$	$3\pi/2$
2	0.5734	0.1331	$3\pi/2$	0
3	0.5888	0.1146	$3\pi/2$	$3\pi/2$
4	0.5696	0.1052	$3\pi/2$	π
5	0.5933	0.1098	$3\pi/2$	$\pi/2$
6	0.6008	0.1230	$5\pi/4$	$3\pi/4$
7	0.5972	0.1217	$5\pi/4$	$\pi/4$
8	0.5853	0.1136	$5\pi/4$	$5\pi/4$
9	0.5683	0.1181	$5\pi/4$	$5\pi/2$
10	0.5892	0.1186	$5\pi/4$	π
11	0.5786	0.1239	$5\pi/4$	$7\pi/4$
12	0.5668	0.1299	0	0

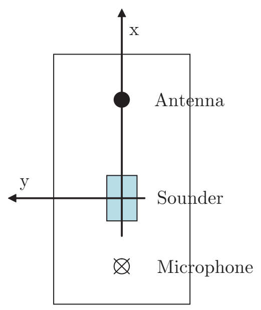

Figure D.1: Inter-node ranging technique: experimental results.

The bias and the standard deviation describe the uncertainty in the observing process. Several are the sources of such uncertainty. First of all, the parameters used to characterize the propagation velocity of an acoustic wave in the air have been considered fixed, while they change according with humidity and temperature. Secondly, the transmission protocol introduces a delay, which cannot be taken into account, as it is not directly observable.

Appendix E

The Paths Construction Problem as Integer Linear Programming

E.1 Simple Partitioning

Given a graph $G = (N, L)$ with N the set of nodes (with cardinality n) and L the connectivity matrix (with cardinality l), $H \in N$ the set of sinks (with cardinality k) and R the set of partition's indexes (always with cardinality k). The simple partitioning problem can be formulated as follows:

$$min \qquad 1 = 1$$

$$s.t. \qquad \sum_k \sum_i p_{ij}^h = 1 \qquad \forall j \in N \setminus H \tag{E.1}$$

$$\sum_k \sum_i p_{ij}^h = 0 \qquad \forall j \in H \tag{E.2}$$

$$\sum_k \sum_i p_{ji}^h = 1 \qquad \forall j \in H \tag{E.3}$$

$$\sum_{i \in H} \sum_{j \in \mathcal{N}(i)} p_{ij}^h = 1 \quad \forall h \in R \tag{E.4}$$

$$\sum_{h \in \mathcal{N}(j)} p_{jh}^h \leq \sum_{i \in \mathcal{N}(j)} p_{ij}^h \qquad \forall j \in N \setminus H, \quad \forall h \in R \tag{E.5}$$

$$\sum_k p_{ij}^h + p_{ji}^h \leq 1 \qquad \forall (i,j) \in L \tag{E.6}$$

$$\sum_{(i,j) \in L \cap V} p_{ij}^h \leq |V| - 1 \qquad \forall V \subset N \setminus H, V \neq \emptyset,\ \forall h \in R \tag{E.7}$$

$$\left\lfloor \frac{N}{k} \right\rfloor - 1 \leq \sum_{(i,j) \in L} p_{ij}^h \leq \left\lceil \frac{N}{k} \right\rceil - 1 \qquad \forall h \in R \tag{E.8}$$

where,

$$p^h_{(ij)} = \begin{cases} 1 & \text{If the link } (i,h) \text{ belong to the path } h, \\ 0 & \text{otherwise.} \end{cases} \tag{E.9}$$

In detail, the (E.1) constrains each node to belong only to one path, the (E.2) and (E.3) are boundary conditions for the heads. The (E.4) forces the heads to belong to different paths, the (E.5) is similar to the *flow conservation* constraint of network flows problem. Together with constraint (E.1), it forces each node $j \in N \setminus H$ to have at most one outgoing link (note that sources will have only an incoming link). The (E.6) avoids the paths to go backward, the E.7, known in literature as *Sub-tour elimination constraint* avoids to have isolated cycles. Finally the (E.8) implies that all paths will have either length $\lfloor \frac{N}{k} \rfloor$ or $\lfloor \frac{N}{k} \rfloor + 1$.

The bogus objective function points out that any feasible solution is inherently optimal for this formulation. If the distance between adjacent nodes is not considered constant, this formulation can be slightly modified to account the minimization of the overall length of the paths. Additionally, note that this problem might not be feasible if the connectivity matrix is not carefully chosen. However, for a sensor network this problem can be easily overcome by properly tuning the range of connectivity among nodes.

E.1.1 Shortest Paths Partitioning

In order to minimize the overall length of the paths, the simple partitioning formulation can be modified by introducing a distance matrix D of Euclidean distances between pairs of nodes along with a real objective function.

$$\begin{aligned}
min \qquad & \sum_{k}^{R} \sum_{(i,j)\in L} d_{ij} \cdot p_{ij}^{k} \\
s.t. \qquad & \sum_{k} \sum_{i} p_{ij}^{k} = 1 \qquad \forall j \in N \setminus H && \text{(E.10)} \\
& \sum_{k} \sum_{i} p_{ij}^{k} = 0 \qquad \forall j \in H && \text{(E.11)} \\
& \sum_{k} \sum_{i} p_{ji}^{k} = 1 \qquad \forall j \in H && \text{(E.12)} \\
& \sum_{i\in H} \sum_{j\in \mathcal{N}(i)} p_{ij}^{k} = 1 \qquad \forall k \in R && \text{(E.13)} \\
& \sum_{h\in \mathcal{N}(j)} p_{jh}^{k} \leq \sum_{i\in \mathcal{N}(j)} p_{ij}^{k} \qquad \forall j \in N \setminus H, \quad \forall k \in R && \text{(E.14)} \\
& \sum_{k} p_{ij}^{k} + p_{ji}^{k} \leq 1 \qquad \forall (i,j) \in L && \text{(E.15)} \\
& \sum_{(i,j)\in L\cap V} p_{ij}^{k} \leq |V| - 1 \qquad \forall V \subset N, V \neq \emptyset, \ \forall k \in R && \text{(E.16)} \\
& \left\lfloor \frac{N}{k} \right\rfloor - 1 \leq \sum_{(i,j)\in L} p_{ij}^{k} \leq \left\lceil \frac{N}{k} \right\rceil - 1 \qquad \forall k \in R && \text{(E.17)}
\end{aligned}$$

In detail, the cost of a path is defined as the sum of the Euclidian distances between successive nodes belonging to this path.

List of Figures

List of Tables