2010

# Collective cluster-based map merging in multi robot SLAM

Ahmad Soleimani
*University of Windsor*

# COLLECTIVE CLUSTER-BASED MAP MERGING IN MULTI ROBOT SLAM

by

Ahmad Soleimani

A Thesis
Submitted to the Faculty of Graduate Studies
through Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2010

*Your file Votre référence*
*ISBN:* 978-0-494-62734-1
*Our file Notre référence*
*ISBN:* 978-0-494-62734-1

# Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this Thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

# Abstract

New challenges arise with multi-robotics, while information integration is among the most important problems need to be solved in this field. For mobile robots, information integration usually refers to *map merging.* Map merging is the process of combining partial maps constructed by individual robots in order to build a global map of the environment.

Different approaches have been made toward solving map merging problem. Our method is based on transformational approach, in which the idea is to find regions of overlap between local maps and fuse them together using a set of transformations and similarity heuristic algorithms. The contribution of this work is an improvement made in the search space of candidate transformations. This was achieved by enforcing pair-wise partial localization technique over the local maps prior to any attempt to transform them. The experimental results show a noticeable improvement (15-20%) made in the overall mapping time using our technique.

# Dedication

*This thesis is dedicated to my beloved parents whom I owe every moment of joy and success in my life.*

*Also, this thesis is dedicated to my loyal wife, Zeinab who did not hesitate to put most of the family responsibilities on her shoulders during my studies while at the same time the smile never disappeared from her face.*

*Finally this thesis is dedicated to my lovely twin Angels, Yasamin and Malaak whom their "dadi" calls and "jump for hugging" used to take out the tiredness from my body and soul at the end of every long working day*

# Acknowledgements

*First of all, I would like to thank my supervisor, Dr. Dan Wu, for his continuous assistance and advices, and also for his encouragement and understanding during the entire period of Master study, especially this thesis work.*

*Besides, I would also like to thank my external reader, Dr. Chunhong Chen, my internal reader, Dr. Jessica Chen, and my thesis committee chair, Dr. for spending their precious time to review this thesis and putting down their valuable comments and suggestions on this work.*

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| EKF | Extended Kalman Filter |
| IR | Infra Red |
| LRF | Laser Range Finder |
| MRDS | Microsoft Robotics Developer Studio |
| MVS | Microsoft Visual studio |
| PDF | Probability Distribution Function |
| RBF | Recursive Bayesian Filter |
| SLAM | Simultaneous Localization And Mapping |
| USAR | Urban Search and Rescue |

# Chapter 1

# INTRODUCTION

This thesis addresses the problem of *map merging* in multi-robot environments. Map merging is the process of combining partial maps built by individual members of a team of robots or multiple runs of a single robot in order to obtain the global map of the environment in a shorter time and higher coverage of the mapping area.

Exploring an unknown environment and constructing its map using mobile robots is a well-known problem in the field of Artificial Intelligence (AI) and robotics. It has been studied widely during the last two decades, using a single robot equipped with different kinds of perception sensors and important successes have been achieved [23, 48]. Recently, most of the research in this field has turned to focus on using multiple or team of robots. If fact, multi-robotics is aimed to fulfill the increasingly demand for automation of difficult tasks and high risk missions, such as planetary exploration, scouting, rescue operations in catastrophe conditions, cleaning, etc. In such environments, the complete coverage of the terrain is a result of the integral parts of a multi-robotic mission [9].

Consequently, autonomous mapping could benefit more from deployment of cooperative multi-robot systems. A team of robots would have a higher degree of perception of the environment due to a larger number of sensors, potentially heterogeneous ones, being used in the area where robots are performing their mapping task [7]. A well-designed team of robots can considerably reduce the time required to map a given environment, since the process of mapping different parts of the environment can be done in parallel. In addition, the overall mapping task using multiple robots is more robust, since the failure of one of the mapping agents will not lead into an entire failure of the mission. In order to maintain this robustness, distributed functionality is a must. Hence, each robot has to perform

its task completely autonomously and independently while there should not be an agent with unique software or hardware features making it vital for the success of the mission [7]. Also, a multi-robot system must benefit from a high scalability level, in which adding a new robot or removing existing one should not require huge amount of reconfiguration and/or restructuring operations [7].

It is evident that the purpose of having a multi-robot system is to have them achieve the assigned task more reliably and in a shorter time. However, the main challenge of such a system is how to put together and effectively combine the data acquired by individual robots. In fact, this is what is called the *problem of map merging* in the field of robot mapping. This thesis work is intended to investigate this problem and it comes up with a considerable enhancement to an existent method of map merging for environment represented by occupancy grids.

In particular, our approach to the problem of map merging is similar to the work of Birk, A. and Carpin, S. [7]. In this work along with some others [10, 12, 35], map merging is based on finding similarities in the grid representation of the local maps built by each individual robot and then by applying geometric transformations, a best match (overlap) between them is being used in the merge process. In other words the target is to find the transformation which provides the maximum overlap between local maps.

It is clear that this approach-beside its benefits, involves dealing with a huge search space of possible transformations which could negatively affect the real-time nature of the system. Therefore our proposed method is aimed to reduce the search space among the possible candidate transformations without losing the accuracy and effectiveness of this approach. For this reason, we try to improve this algorithm in two aspects: Firstly, upon a meeting event between two robots, we will not perform any transformation unless the other robot is partially localized in the first local map, since otherwise, the possibility of getting a false positive transformation (that believed to be a true matching, while it is in fact a false one) will be high. Secondly, only those transformations located within a Gaussian distribution with mean $\mu_c$ and covariance $\Sigma_k$ around the point of meeting

(C) will be considered. In this notation, $k = \alpha.grid_{size}$ is dependant to the grid size of the map. This assumption is being made in order to insure dismissing as much unwanted transformations as possible from our search space, considering the fact that the overlap region is most probably located somewhere around the point of meeting. The results from experiments showed that by applying these improvements we managed to increase the transformation effectiveness (a metric to measure the rate of desired transformations) from 71% in Basic multi-robot SLAM to 86% in our proposed SLAM method.

The rest of this document is organized as follows. Chapter 2 provides a background study about robot localization and mapping. Chapter 3 reviews the related literature in the field of map merging. The fundamental part of this document is Chapter 4, where our approach to the problem of map merging is being described and the proposed method is presented. Chapter 5 illustrates the results obtained from different experiments conducted to test the functionality and performance of our proposed method. Finally in Chapter 6 we outline the conclusions and mention some possibilities for future work.

# Chapter 2

# BACKGROUND STUDY

This chapter briefly illustrates the relevant topics to robot's map merging problem. It provides the necessary background in order to present our map merging proposed method. Section 2.1 briefly talks about the topic of *robot mapping*. Section 2.2 describes *occupancy grids* and *topological* maps-the two most popular methods of map representation in the field of robot mapping. A mobile robot needs to keep track of its location (be localized) in order to perform its assigned duties, a subject covered by Section 2.3. Finally section 2.4 briefly explains the well-known method of Simultaneous localization and mapping (SLAM)

## 2.1 Robot mapping

"Robot mapping is that branch of one, which deals with the study and application of ability to construct map or floor plan of the environment by the autonomous robot and to localize itself in it" [24]. In fact, for any mobile agent including humans, in order to perform any task or mission which requires relocation, a prerequisite necessity will be to know the distribution of the occupied and non-occupied spaces or in other words, the "map" of the environment. This knowledge enables the mobile agent to perform the assigned tasks appropriately. Therefore if the map of the exploration environment is given and the mobile agent is aware of its position within that map (localized), it can use its sensor readings data along with the motion model of the environment to achieve its goals.

On the other hand, however, the knowledge about the environment structure (map) is usually not available to the mobile robot as a *priori*, and must therefore be acquired through the sensors of the robot and used along with the robots

motion model to build a partial or complete map of the environment. Furthermore, this constructed map needs to be properly represented in order to be used effectively by mobile robots.

## 2.2 Map representation

There are different methods used to represent the maps of the environments. In mobile robotics field, however, two of these representations, *occupancy grids* and *topological*, are the most popular methods used for the purpose of robot exploration.

### 2.1.1 Occupancy grid maps

Occupancy grids are the most popular method used to represent the map of the environment of a mobile robot when there is no prior information about the physical structure of the environment. In this method the environment is represented by a grid of cells in which a cell is either filled (part of an obstacle), empty (part of a free space) or unknown. Each cell holds a probability that the cell is occupied by an object while the attributes of that object (shape, color, etc.) do not make up a concern for the representation process. Grid maps are useful for combining different sensor scans, and even different sensor modalities- Sonar, laser, IR, bump, etc. [47]

In fact, this kind of representation is of particular interest when the robot is equipped with LRF sensors [1]. In this case, for each cell a counter might be considered in which the value of zero indicates that the cell has not been hit (visited) by any ranging measurements and hence is likely to be free space. As the number of hits increases, the cell's value is incremented and over a certain threshold the cell is considered to be occupied (by a possible obstacle). On the other hand, the values of the cells are usually decremented when a ranging beam travels through the cell, striking a further cell. This approach can be

extended to support transient obstacles by decrementing the cell values over time. [28]

Figure 2.1 is an example of occupancy grid representation for a simple environment which includes three enclosed objects with uneven surfaces. In this representation, black cells indicate obstacles; white cells indicate free spaces and grey cells show uncertainty status. One of the drawbacks of this representation is the high amount of memory consumption used to store the mapping data, since the size of the map in robot memory is directly related to the size of the environment.

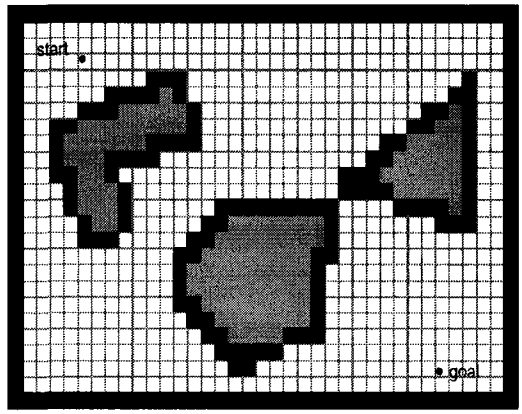


*Figure 2.1: occupancy grid map. Black cells represent obstacles, white cells represent free spaces and grey cells show uncertainty status [28]*

## 2.1.2 Topological maps

Another popular representation for robot maps is the topological (landmark-based) representation. It avoids using the actual measurement of geometric attributes of the environment, but focuses instead on the landmarks (robot recognizable objects or features of objects) of the environment. A topological map can be built using a graph data structure in which the nodes capture the objects and edges represent the connectivity between those objects. When an edge connects two nodes, the robot can traverse from one node to the other without passing through any other node (object). Figure 2.2 shows a topological

representation for a small office area. In this representation, the nodes of the graph (1 to 18) demonstrate the landmarks while the edges denote the connectivity (reachable paths) between them.

It is evident that the amount of memory space needed to represent a map using this method is much lower than the corresponding grid representation. Therefore, the compactness of this representation beside the ability to embed the non-geometric features used for localization process are the main advantages of this method. However the lack in expressiveness of the robot position in an accurate way makes it inappropriate for the task of mapping where a detailed expression of the environment is needed [28].

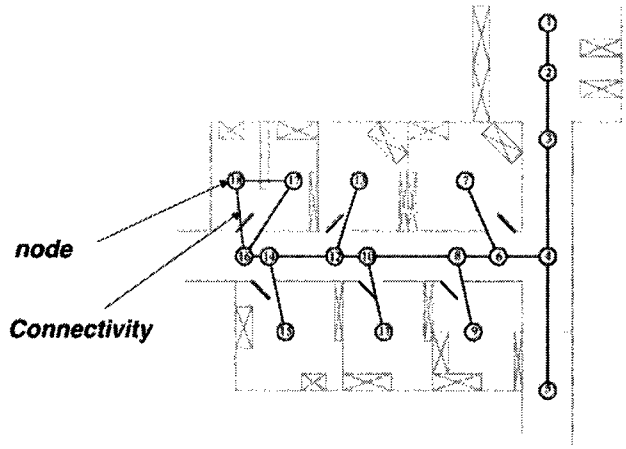


*Figure 2.2: a topological representation of an indoor office area. In the above graph, each node denotes an individual landmark (from 1 to 18) and each edge denotes the connectivity between two landmarks [28]*

## 2.3 Robot localization

For a mobile robot in an environment with known map, the very beginning step in order to perform an assigned task is to be aware of its current position in the map of the environment. In other words a mobile robot needs a reliable position estimation mechanism to navigate precisely in the environment and consequently fulfill the assigned missions and tasks.

In fact, the main problem of robot localization can be break down into two sub problems of global position estimation and local position tracking [28]. Global position estimation is to specify the position of the robot in a given map and once the robot is localized we need to keep track of the robot position over time. To be able to perform a successful mission, providing both of these capabilities is essential.

There are different approaches to solve the problem of robot localization problem. However, considering the probabilistic nature of this problem which is due to the uncertainty and noise associated with sensor readings and also motion model of mobile robots, it would appear that probabilistic approaches are among the most adequate candidate methods to provide a comprehensive real-time solution to this problem. Therefore, methods based on Bayesian reasoning approach have attracted most of the research in situations where the environment is represented by occupancy grids with no landmarks [13].

The Bayesian model approach provides the general framework for the estimation of the system state (robots poses) in the form of a probability distribution function. In fact, Bayes filter recursively computes the posterior probability over poses and partial map given previous sensor measurements & motion commands. Figure 2.3 illustrates the recursive interaction between the robot pose $(x_t)$, an observation made by the robot $(z_t)$ and a robot motion command $(u_t)$ all given at time *t.*

The recursive Bayesian filter (RBF) is a probabilistic framework for state estimation that utilizes the Markov assumption (i.e., past and future measurements are conditionally independent, if the current state is known) [17]. The RBF estimates the posterior belief of the robot position given its prior belief, motion and sensor measurements, and the model of the world (or environment). A *belief* reflects the robot's internal knowledge about the state of the environment. [47]

In particular, the prior belief is a probability distribution over all possible locations before taking the motion actions and sensor measurements into account. The posterior belief is the conditional distribution of these locations after

incorporating the motion actions and sensor measurements. As shown in Table 2.1, the belief about the robot pose at time $t$, $bel(x_t)$, is calculated from the belief $bel(x_{t-1})$ at time t-1, the last motion action $(u_t)$ and the most recent sensor reading $(z_t)$. In fact, this update operation to determine the pose of the robot is being applied recursively to obtain the belief $bel(x_t)$ from the belief $bel(x_{t-1})$ which was obtained in a similar process in a previous iteration of the algorithm.

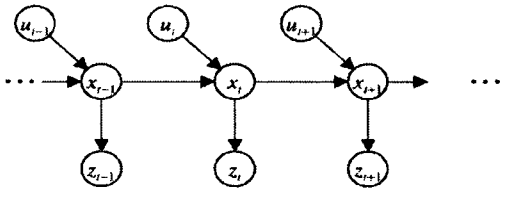

*Figure 2.3: Bayes filter recursively computes posterior probability over Poses and partial map given previous sensor measurements & motion commands. Here $x_t$ stands for a robot pose, $z_t$ an observation made by the robot, and $u_t$ a robot motion command given at time $t$. [9]*

**Algorithm Bayes_filter**$(bel(x_{t-1}), u_t, z_t)$:

$for\ all\ x_t\ do$

$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}$

$bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$

$endfor$

$return\ bel(x_t)$

*Table 2.1: General algorithm for Bayes filtering [47]. the belief about the robot pose at time t, $bel(x_t)$, is calculated from the belief $bel(x_{t-1})$ at time t-1, the last motion action $(u_t)$ and the most recent sensor reading $(z_t)$.*

This approach along with particle filter technique builds the kernel of the widely used *Monte-Carlo* localization method. In particular, based on this approach, a set of samples randomly drawn from the probability density, will be used in an iterative process of matching the sensor data at the current position with the existent model of the environment and a consequent update is made to

the probability distribution function based on the results obtained. This sampling-based localization algorithm is based on a three step process of prediction, re-sampling and update tasks and will be running continuously until a certain threshold of accuracy as a success indicator is met. (For more information on the algorithm of Monte-Carlo and particle filter approach, refer to [47])

It is clear that in the event of absence of the environment map, the process of robot localization and tracking loses its sense since a localization task is being performed to specify the pose (position and orientation) of a mobile robot over time in a well sketched (mapped) environment. In fact, in real environments the problem of localization and mapping usually appear together when the robot is placed randomly in an unknown area. According to this assumption, a simultaneous task of mapping and localization can provide a one-pass reliable solution for both problems. This approach has attracted a lot of research in the field of robot localization and mapping and established the well-known method of SLAM (Simultaneous Localization And Mapping).

## 2.4 SLAM

Simultaneous localization and mapping (SLAM) is a fast growing method that has attracted many researchers in the field of autonomous map building. We should bear in mind that SLAM is not a specific algorithm but rather it is a technique and conceptual approach [4]. SLAM addresses the problem of building a map of an unknown environment by a mobile robot while at the same time keeping track of the navigating robot in the partial map being generated gradually by the robot. In fact, the processes of robot locating and mapping under conditions of errors and noise do not allow for a straight-forward solution of both operations. Therefore, SLAM is an approach to bind these processes together and to create a mutual interactive mechanism which iteratively interchanges feedbacks from one process to the other in order to enhance the results of both consecutive tasks [47].

Therefore, the goal of SLAM is to use the environment data (through the sensor readings) to continuously discover the distribution of the free and occupied spaces in the exploration environment (mapping part) while keep updating the belief about the real location of the mobile robot within the local constructed map (localization part). This process is being performed by adding newly observed parts of the environment to the previously visited regions.

In particular, the odometry data measured by the movement of the wheels of the robot feeds the SLAM algorithm with an initial approximation of where the robot might be, and then a correction process using the readings from the robot sensors along with the motion model rules will be applied to that initial approximation to derive an accurate estimate of the real position of the robot within the local map. In fact, since the posterior distribution of the SLAM is related to the current and all previous sensor readings ($z_{1:t}$) and also to all but current motion commands ($u_{0:t-1}$), it can be written as $p(x_{1:t}, m|z_{1:t}, u_{0:t-1})$ in which all time poses of the robot ($x_{1:t}$) along with the map ($m$) have to be determined.

Considering the fact that map $m$ is independent from the motion actions of the robot, the above conditional probability can be expanded to a product of two conditional posteriors of $p(x_{1:t}|z_{1:t}, u_{0:t-1}).\ p(m|x_{1:t}, z_{1:t})$. Therefore the problem of simultaneous localization and mapping will be turned into two sub-problems of *partial localization* and *mapping with known poses*. Figure 2.4 illustrates the general probability distribution formulation of the SLAM problem and the mutual interaction between the robot pose ($x_i$), odometry data ($u_i$), sensor readings ($z_i$) and partial map (m) .

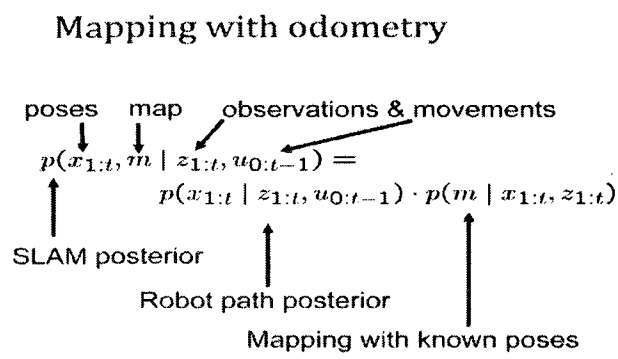


*Figure 2.4: General formulation of SLAM problem using odometry data. Considering the independence between the map and the motion actions, the LHS can be rewritten as a product of conditional probability distributions. This action in fact reduces the complexity of SLAM problem into problems of partial localization and mapping with known poses.*

Chapter 3

# MAP MERGING

The aim of this chapter is to provide the reader with a literature review of the recent important works done in the field of map merging. The first section gives an overview of the map merging problem, while the second section classifies the associated literature and current map merging methods into three groups based on their interaction with the issues of robots *odometry*, *intercommunication* and *initial poses*.

## 3.1 Overview

Cooperative exploration and building a reliable model of the environment in multi-robot systems is a key criterion to assure the autonomy of the system [1]. In a typical multi-robot system, the participating robots build maps in their local coordinate systems which need to be transformed into a global coordinate system. The procedure of estimating these transformations and fusing the locally created maps together in order to build a joined global map is known as the map-merging problem. In many approaches the problem is being considered as a search problem by repeatedly proposing candidate transformations and verifying the quality of them. The differences are differentiated by metrics guiding their proposal and verification processes. [1, 39]

The first developed multi-robot exploration system was a simple extension of the single robot implementation [4, 15]. It is clear that these systems are more sophisticated than other distributed systems due to the difficulty in modeling a real world environment in which the entities are dynamic and unpredictable besides the fact that sensor readings are noisy and motion model information is inaccurate. In fact, putting together multiple robots in an environment brings a

new set of challenges and difficulties in robot coordination, inter-communication and data integration.

Data integration deals with the techniques of combining distributed partial information collected by different robots operating in several parts of the environment. Data integration can happen at different levels but in this context, the sub-problem of how to merge local maps built by several robots into a unified global map, forms the essence of the problem. According to [28], this problem of map merging, "Is an interesting and difficult problem, which has not enjoyed the same attention that localization and map building have". During the last few years, this issue has attracted an increasing number of researchers in the field of multi robotics and multi-agent programming and consequently has resulted in developing important practical algorithms and techniques to best deal with this challenge. Map merging plays a crucial role in multi-robot SLAM (Simultaneous Localization and Mapping) and USAR (Urban Search and Rescue) fields.

Approaches to map merging were made from different perspectives and set of assumptions resulted in several techniques and strategies in dealing with this problem. Among these main perspectives are, existence of real time communication between the agents [18, 24], prior partial knowledge about the map of the environment, knowledge about the initial location of participating agents [44, 24], and using of odometry information [40, 33]

Furthermore, the way the environment is represented plays an important role in setting up the appropriate method of map merging. Therefore, another classification of map merging methods exists based on whether the maps are represented with their features (landmarks) or based on occupancy grids. In a feature based map only distinguishable landmarks of the environment will be registered in the map and eventually considered for the later map processing procedure while in the grid-based representation, each specific piece of area (grid cell) of the environment will be dealt with and assigned a posterior probabilistic value of being occupied or free based on sensors perception.

## 3.2 Classification and literature review

Multi-robot systems and multi agent programming are the most recent state-of-the-art topics in the field of Artificial intelligence and robotics. Map merging is the process of building a consistent model of the environment using the data collected by several robots [35]. If the initial positions of the robots are known, map merging becomes a simple extension of a single robot mapping. In contrast, if the robots do not know their relative positions, the problem becomes more difficult; as it has to be determined how and where individual robot perceptions of the environment (local maps) should be integrated into a comprehensive global map. As mentioned before, some assumptions about robots, and also the availability or absence of some important data, play an essential role in specifying the kind of approach and possible solution for the problem of map merging. The following sections describe briefly the classification of map merging techniques based on these criteria.

### 3.2.1 Odometry information

Odometry is the use of data from the movement of actuators to estimate change in position of the robot over time [24]. The word odometry is composed from the Greek words hodos (meaning "travel", "journey") and metron (meaning "measure").

Whether the robot is wheeled or legged, its position can be estimated through special calculations on the movements of its actuators. In fact, to use odometry effectively, there needs to be an accurate and real time data collection, equipment calibration, and suitable processing resources for storage and analysis of the collected data. Considering map merging, some researchers believe that using odometry in mapping and map merging process can improve the time required and also the accuracy of the final global map.

In particular, the idea of methods based on odometry information is to deal with map merging as a maximum likelihood estimation problem that can be solved by performing a scan matching process aimed at finding the most likely

global map given the data of local maps collected through individual robots. To deal with the problem of errors dependency measurements, Lakaemper *et al.* [34] suggest using a new map merging process based on geometric local process of line segment merging with a global statistical control. On the other side, in order to prevent running into high dimensionality, they propose using a higher level objects presentation (line segments and generalized polygons) to represent the landmarks of the environment, further more they use a process called Discrete Segment Evolution, that minimizes the number of line segments required to represent the mapped environment. Furthermore, to overcome the problem of correspondence, they propose using a sophisticated shape similarity relation method.

In the work of Leon *et al.* [35], the process of building a map from the odometry data obtained by multiple robots is being done by using Scan-match technique. For this purpose they have used GMapping and GridSLAM algorithms obtained from OpenSlam.org with some modifications to the original codes and parameters.

Panzieri *et al.* [40] use the odometry data gathered by multiple robots in a different approach, as they combine topological and occupancy grid map representations in their map merging proposed method. First, an Extended Kalman Filter (EKF) in a SLAM scheme is performed to improve the odometry information and to build a geometric map with specified locations of the beacons and robot poses. Second, considering the fact that fuzzy theory models the sonar reading uncertainties better than a probabilistic approach, they create a Fuzzy Global Map (FGM) using ultrasonic range finders while navigating the environment. Then the FGM can be applied to the estimation to increase the accuracy of the maps by comparing the mapping area with a fuzzy local map (FLM). Finally a special artificial visual perception (FLM vision) of the robot environment is computed from the actual sonar data and used for topological localization.

In some other approaches, the idea of using odometry data is completely dismissed. Amigoni *et al.* [4] justify not using odometry data to be able to

interrupt the mapping process and resume it at a later time without having to reset the poses of the robots, which gives them a solution for the kidnapped robot problem. With this approach, a three step procedure is being performed in which, first appropriate transformations based on the angles between line segments of the scans are being found, then the related transformation are being evaluated and finally the best found transformation are being applied and the scans to build pairs of scans. To create the final general map, theses pairs of scans need to be merged using special methods.

Carpin [11] ignores the usage of odometry data gathered by participating robots and in turn bases his approach on finding the best transformations (rotations then translations) made on one map to best overlap with the other maps creating the general map of the environment. The transformations are weighted using an "acceptance index" function defined based on the number of matching cells – free or occupied – in the merging maps. To specify the best transformation, a metric is necessary, while in many of works done using this approach, Hough Spectrum analysis is being used to solve the problem of scan matching. Hough transform is being used to detect lines and other geometric curves that can be parameterized with few values. As Hough spectra are one-dimensional signals, cross correlation between two such signals can be used to determine similarities and therefore finding the best overlapping transformations. The main purpose of this paper is to develop a method to perform the task of map merging in the field of urban search and rescue robotics where multiple robots are involved in the search and rescue task. In the begging of this research, the authors mention some of the difficulties related to this kind of problem and summarized them as lack or inaccuracy of reliable odometry information which results in unknown poses of the robots, missing distinguishable landmarks due to the catastrophic scenarios and finally minimal scan overlap of the robots sensor data.

It is clear that map merging in the field of urban search and rescue robotics where multiple robots are involved in the search and rescue task in an uneven environment is a complicated task in which the odometry information are not

very reliable due to the conditions of the environment. Lakaemper *et al.* [32] develop a method to perform the task of map merging in the field of urban search and rescue robotics where multiple robots are involved in the search and rescue task. In this method, a sophisticated data structure is being created and then optimized to represent local maps in term of graphs and they use a technique called "Force Field Simulation" (FFS) to perform map merging process. This approach is inspired by simulation of dynamics of rigid bodies in gravitational fields in which the physical laws were replaced by human perception constraints.

### 3.2.2 Robots intercommunication

It is clear that having a kind of communication between the members of a team of robots exploring the environment can be beneficial to the process of map merging in the sense of cutting the time required for the entire procedure by eliminating some of the redundancy caused by repeated visits of different robots to the same areas of the environment. However, using communication brings to the scene a new set of challenges. Unreliable continuous wireless communication, due to noise and obstacles, data loss and corruption, data consistency, cooperation and coordination of participating robots, are among some of the new issues that arise in such an environment. Therefore some researchers prefer to include complete or partial communication in their approaches to map merging.

Konolige *et al.* [28] try to turn the problem of mapping and map merging to a decision problem in which each robot is capable of mapping the environment by its own but is also capable of communicating with other robots. This distributed approach is enabled by pair wise relations between participating robots. The interactions between robots can be categorized into No interaction (the robots are not within communication range), Hypothesis generation (the robots are able to communicate but are not aware of their relative locations), Hypothesis verification (robots can communicate and verify the determined location hypothesis) and coordinated exploration (robots can share maps and perform

different levels of coordination) types. The entire multi robot system is summarized by a graph structure in which the nodes are the navigating robots and the edges represent the current interaction between pairs of robots.

Pan *et al.* [39], on the other hand use the intercommunication between robots to build a global map of the environment through merging local maps built by multiple robots using a distance transform and an improved genetic algorithm. The idea of using a genetic algorithm in the field of multi robotics is to use the concept of consecutive generation similarities in finding the maximum overlap at which the local maps can be merged at. The idea behind the genetic algorithm is to generate many individual solutions randomly to build an initial population and then to select a portion of the existing population to generate the second generation population through applying a set of genetic operators such as crossover and mutation. This process continues until a specific threshold is met. To overcome the low speed and search deficiency of traditional genetic algorithms, some techniques such as distance transformation and adaptive strategy genetic algorithm can be adopted to enable these algorithms to search and find the best transformations between the constructed local maps to generate the final global map in a reasonable time.

In the work of Fox *et al.* [18], the communication between participating robots take a wider spectrum and includes the initial poses of the robots as well In particular, the exploring robots start from unknown locations and gradually the robots detections are being integrated into a Bayesian decision-theoretic exploration based map. In such a strategy, each robot explores on its own, mapping a portion of the environment. As soon as two robots can communicate, they exchange sensor information and estimate their relative locations. Once they have a good hypothesis for their relative location, they actively verify this hypothesis using a Rendezous technique. In case of matching, the robots create an exploration cluster: they combine their data into a shared map and start to coordinate their exploration actions. On the other hand, if the relative location hypothesis faces a mismatch, the robots continue to explore independently and exchange sensor data to refine their estimates of their locations. During

exploration, the size of exploration clusters increases as more robots determine their relative locations, ending in a single cluster of all robots.

In contrast, in the work of Birk and Carpin [7], the robots do not interact with each other while drawing the local maps, the resulted maps then will be merged using a special similarity metric (acceptance indicator) and a stochastic search algorithm. In this approach, a multi stage mapping algorithm is being used in which in the first stage, the robots start building local maps independently and represent them in the form of occupancy grids rather than topological- to overcome the issues related to feature based maps, while in the second stage a stochastic transformations search algorithm (Adaptive Random Walk) is used to perform an initial pair-wise merge process based on the highest degree of overlap (maximum acceptance indicator) for every pair of the local maps. Finally, an optimization heuristic function will rearrange and realign the attached portions of the global map.

### 3.2.3 Robots initial positions

The existence of prior knowledge about the initial positions of the navigating robots simplifies and smoothes the process of mapping and map merging by a large degree. In fact, such an assumption eliminates the localization part of the problem; however, it limits the scenarios that can be handled by such a navigation strategy due to its unrealistic hypothesis. A one real-life case of such a condition is where all robots start from the same point.

Stewart *et al.* [44] use the initial poses of the robots to develop a hierarchical Bayesian method which captures the structure of the environment through a hidden Markov process that represents transitions between different views of the area being mapped. A non real-time learning process takes a set of maps and creates a Dirichlet priori over map structures. This priori will be used by the exploring robot as a generative map at the start of the mapping process. Gradually and as the robot encounters views of the environment, an adaptation process will refine the model distribution in real time.

Howard *et al.* [23] assume that all robots start from the same point. They extend the work of Hahnel, *et al.* [2003] in which a single-robot Rao-Blackwellized particle filter was considered and tries to generalize it to handle multi-robot case in which the initial position of the participating robots is known. In this approach, an approximation is being used to solve the more realistic problem of multiple robots in which the initial position of the robots is not known (robots start from widely spread locations) and also to integrate observations collected before robots encounter each other using the notion of virtual robot travelling backward in time.

In contrast, in the work of Wang *et al.* [48], the problem of multi robot localization and mapping is being covered when there is no information about the initial locations of the robots at all. They reformulate the problem of multi-robot SLAM into a mapping static estimation problem by having the locally built maps being fused into a jointly maintained global map through decoupled SLAM (D-SLAM) framework. Since the exploring robots are not aware of their initial locations, they start navigating and building a local map by using a traditional extended Kalman filter algorithm and the resulted local maps will be uncorrelated to each other. The alignment of these local maps is being performed by special algorithm inspired by the method of medical image registration and the test of joint compatibility. At specific intervals, the D-SLAM framework will be used to fuse them into a global map.

Adluru *et al.* [1] assume a complete absence of initial robots poses. They reduce the problem of multi robot SLAM into a SLAM problem for a single "virtual" robot. This approach allows them to adapt the SLAM localization algorithm of Rao-Blackwellized for solving the problem of map merging. They imagine an exploring virtual robot using the individual robots as its sensors and the local maps created by the real robots replace the local scans. The main differences between the proposed method and the single robot SLAM are related to the motion and perception models of the virtual robot.

Chapter 4

# COLLECTIVE CLUSTER-BASED MAP MERGING IN MULTI-ROBOT SLAM

This chapter is intended to introduce our proposed method to solve the problem of map merging in a multi-robot environment. In section 4.1, an overview of developed method along with a brief description about the possible communication interactions between a pair of robots is provided. Section 4.2 illustrates the *overlap* implication in the field of map merging. The formal formulation of the map merging problem along with our grids similarity and transformations based approach is presented in Sections 4.3 and 4.4. Finally in Section 4.5, a mechanism for failure detection in our method is introduced.

## 4.1 Overview

Considering the different approaches to solve the problem of map merging in multi-robot SLAM, our proposed method is based on grid image similarity approach presented by Birk, A. and Carpin, S. [7]

In particular, each robot starts to explore the environment and builds its local map. Initially it is assumed that each robot belongs to an imaginary growing cluster which encloses the local map of the robot. Hence, there will be several mapping clusters equal to the number of participating robots at the beginning of the mapping process. Once a pair of robots meet with each other and exchange their mapping information, their associated clusters will be merged together to form a united cluster enclosing the union of the two local maps. As the process of mapping goes on, these mapping clusters continue to merge gradually and the ultimate goal of the mapping task will be having a single global cluster which includes all the mapping robots and covers the entire area of the environment.

In fact, this approach is a distributed approach in which each robot has the capability of mapping the environment by itself and also at the same time can discover and establish interactive communication intermittently with other "colleague" robots. For each pair of robots, while they are in a mutual communication state, they can exchange and share mapping information and coordinate their exploration tasks. Hence, this approach of mapping and exploration is based on pairwise relations between the members of the robots team.

In particular, considering a pair of cooperating robots, three types of interaction statuses are expected:

1. **No interaction**: The robots do not fall in the communication range of each other. In other words, the mapping clusters of the two robots do not have any intersection regions. Hence they cannot have information sharing or exchange between themselves at this stage. (see Figure 4.1)

2. **Visible**: The robots fall within the sensor range of each other and therefore they can communicate with each other but it is still not guaranteed that they can merge their mapping data. In fact, this is related to the ability of pairwise pose estimation of the robots. If the observed robot can be localized in the partial map of the observing robot then there is a good chance for a successful map merging between the two robot's local maps (merging candidates). Otherwise, the observing robot will ignore this instance of meeting and continue its local mapping (see Figure 4.2)

3. **Merging candidate:** Once the robots specified their relative locations (were localized in the partial maps of each other), a set of candidate transformations will be applied to one local map trying to satisfy the overlap requirements (set by the overlap heuristic algorithm) with the other partial map (local cluster map). In the event of having a successful

transformation, the robots can merge their local maps and establish a new combined mapping cluster which will makes up their belief about the map of the environment at that point of time and consequently continue to perform coordinated exploration.

An interesting feature of this interaction type is the transitivity attribute, *i.e.* if robot *R1* and *R2* can merge their maps and robot *R2* and *R3* can merge their maps too, then all three robots can build a unified partial global map. Therefore, in this case the resulting mapping cluster will include robots *R1, R2* and *R3* and enclosed region of the cluster is the union of the local maps of all three robots. For intercommunication purpose, it is assumed that a wireless communication between the exploring robots is existent. This communication is intermittent and based on time intervals. Also, the initial poses of the robots are completely unknown.

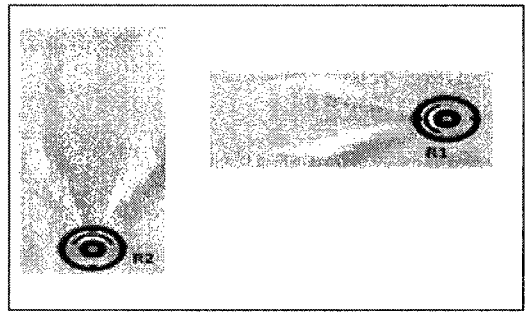


*Figure 4.1: robots R1 and R2 do not fall in the sensor range of each other and their local maps do not have any common regions. No data exchange is possible*

For the task of localizing one robot in the local map of the other, we use the method proposed by Howard [24] for partial map localization in which a modified version of particle filter along with Monte-Carlo algorithm perform the task of localization. (For more information about this method refer to [24] and [47])

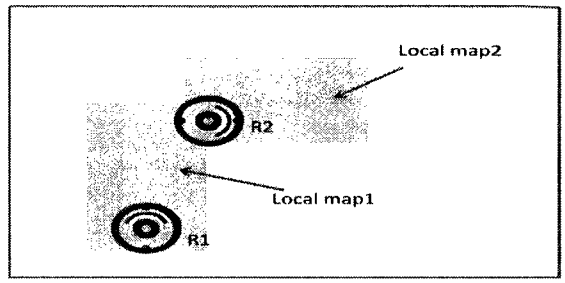


*Figure 4.2: robots R1 and R2 fall within the communication range of their sensors and a possible overlap between the two local maps has to be verified. In case of the availability of an overlap, the two local maps will be merged together to form a partial global map and the two robots build a combined mapping cluster*

## 4.2 Overlap

It is clear that having multiple robots exploring different parts of the environment in the task of mapping can potentially speed up the task of mapping. Initially, each robot starts the task of mapping by its own without any prior knowledge about it's or other's location and builds a partial (local) map of the environment. The main challenge will be how would it be possible to integrate those partial maps into a global map. A popular method used to address this problem is to identify regions of intersection (overlap) between the local maps at which they can be integrated together [4, 7, 15]. To fulfill this aim, a special method for similarity measurement and also a stochastic search algorithm are necessary to enable us to pick up the right set of possible transformations.

In fact, the purpose of overlap is to find the common areas between the two local maps [7]. Since each mapping robot starts its mapping task from a different starting position with different bearing, a possible common area shared between the local maps would not be visible from the first sight and it needs to be discovered. One of the popular solutions to find the regions of overlap is to transform (translate and rotate) one of the maps to overlap (partially cover) the other map. In Figure 4.3, given two maps *m1* and *m2* with regions of overlap (shown by grid shaded areas in part A), the search algorithm transforms *m2* by rotations and translations to find a maximum overlap with *m1* while *m1* is being kept in stationary status during the transformation process (part B). In this

operation, a heuristic similarity method (in our approach, *picture distance function*- to be defined shortly) keeps guiding the search algorithm toward the best solutions [7].

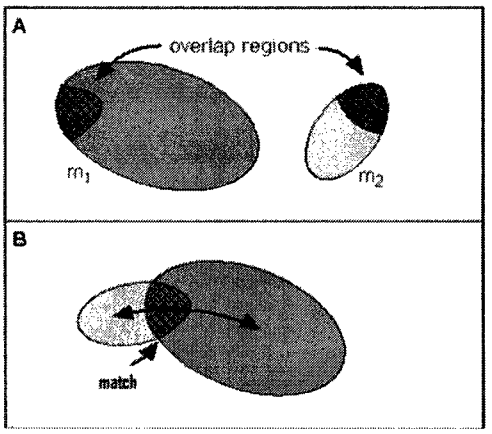


*Figure 4.3: Given two maps m1 and m2 with similar regions (green shaded) in (A), the search algorithm function transforms map m2 by rotations and translations to find a maximum overlap between m1 and m2 while map m1 remains still in this process (B) [7]*

## 4.3 Notation and problem definition

**Definition:** Given two maps $m_1$ and $m_2$, the goal of a merging process between the two maps is to find a transformation *T* so that the two maps can be overlapped. Transformation *T* is a combination of rotation $\psi$ followed by a translation along *X* and *Y* axis of $\Delta x \,\&\, \Delta y$ respectively.
Transformation *T* can be represented as 3x3 matrix:

$$T(\Delta x, \Delta y, \Psi) = \begin{bmatrix} \cos\Psi & -\sin\Psi & \Delta x \\ \sin\Psi & \cos\Psi & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$

We assume that the resulting map of the above transformation is map $m_2'$ , hence:

$$m_2' = Tm_2$$

Therefore, the goal of transformation $T$ is to have the maximum similarity between $m_2'$ and $m_1$ . In other words, map merging is an optimization problem in which it is required to satisfy the following condition:

$$\textit{Maximize Similarity} \quad (m_1, T_{x,y,\theta}(m_2))$$

This optimization task has to be performed over a three dimensional space of two translations and one rotation which is similar to the docking problem studied in computational biology (PPI-protein to protein interactions) solved by higher dimensional search [12].

In fact, since there is infinite number of possible transformations, there should be a kind of metric or mechanism to evaluate the quality of the transformations and subsequently pick up those providing the maximum overlap. This issue is what is going to be discussed in the following section.

## 4.4 How good the Transformation is?

In the previous section, we formulated the problem of map merging between two local maps as an optimization problem aimed to find the maximum similarity between a set of possible transformations of one local map with the other local map. To fulfill our target, an effective heuristic method is necessary in order to pick up and subsequently apply the best transformations with the least costs. In our approach, we will be using an improved version of the basic *Adaptive Random Walk* algorithm proposed by Howard [24] for this purpose. Based on our proposed modified version of adaptive random walk, only a subset of the candidate transformations which meet certain conditions will be considered for the transformation task. The criteria for the selection process are as follows:

- The overlap verification process will be performed only when the other robot(s) was localized in the partial map of the mapping robot.

- Only those candidate transformations falling in an enclosed area of a Gaussian distribution with mean $\mu_c$ and covariance of $\Sigma_k$ around the point of meeting (the position of the observed robot within the local map) will be considered. $C$ is the point of meeting and constant $k = \alpha.grid_{size}$. The size of the gird cell, $grid_{size}$, is an environment dependant variable and $\alpha$ is a constant coefficient determined by the experiment.

In Figure 4.4, a brief illustration of this approach is presented in which, R1 is the robot which was observed by the observing robot (R2) inside R2's local map (local map2). In such a condition, only those candidate transformations for R1's local map (local map1) which are located within the defined Gaussian distribution will be considered for the transformation evaluation process. Finally, those transformations selected in the previous step will be used in the process of transforming R1's local map to best fit with R2's local map.

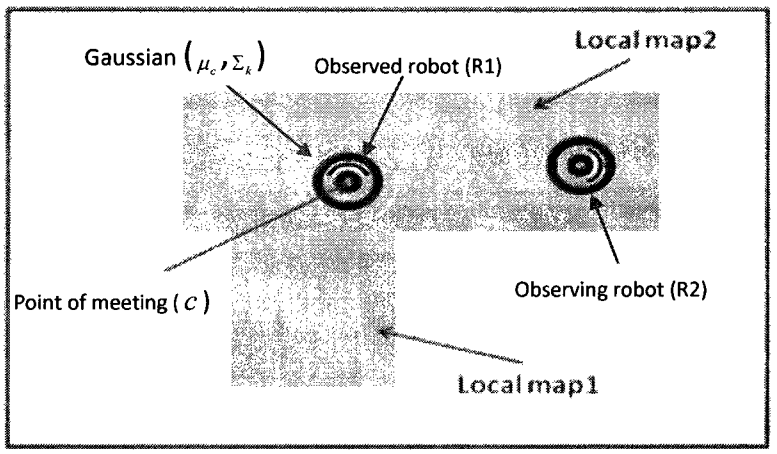


*Figure 4.4: Only those candidate transformations falling in an enclosed area of a Gaussian distribution with mean $\mu_c$ and covariance of $\Sigma_k$ around the point of meeting will be considered. R1 is the robot which was observed by the observing robot (R2) inside R2's local map (local map2). In such a situation, only those candidate transformations for R1's local map (local map1) which are located within the defined Gaussian distribution will be considered for the transformation evaluation process in order to transform R1's local map to best fit with R2's local map*

As mentioned before, we represent our mapping data using occupancy grids,

in which an occupied cell is represented by value "1" while free cells are represented by values of "0" (see Figure 4.5).

In fact, initial grid matrices are not very efficient representation for the task of pattern lookup required in robot map transformation, since it is very possible that it would generates *false positive* transformation candidates. A false positive is a chosen transformation believed to be true matching one, while in fact it is false one and would generate a wrong overlap. The high number of false positives using grid matrices is due to the limited number of distinct values (only 0's and 1's) being used in this representation which in turn causes an increases in the possibility of a false random pattern match. Therefore, a more efficient representation for this purpose which maintains the description of the free and occupied cells patterns is necessary. A distance map can be a suitable alternative to provide such representation and also to reduce the computation complexity [7]. In a distance map, a distance computation function calculates the distance using a pattern recognition distance function (i.e. *Manhattan, Euclidean* or *City block*) between the free cells and the occupied cells.

In particular, occupied cells get a value of zero (since they have a distance of 0 to the nearest occupied cell which are themselves), while other cells will be assigned a value which is equal to the distance (expressed in the number of cells) to the nearest occupied cell (see Figure 4.6). The general formulation of this approach is as follows:

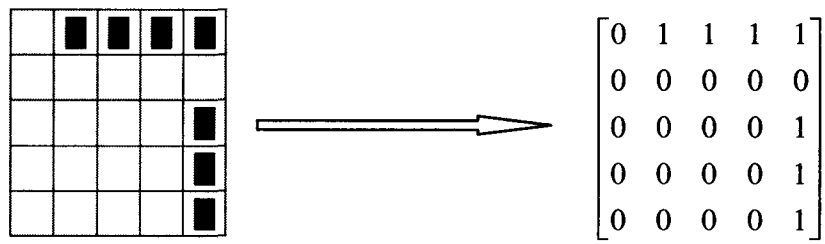


*Figure 4.5: Simple initial grid represented map along with its equivalent grid matrix. Each occupied cell (obstacle) is valued at 1 while each free cell is valued at 0 in the related grid matrix.*

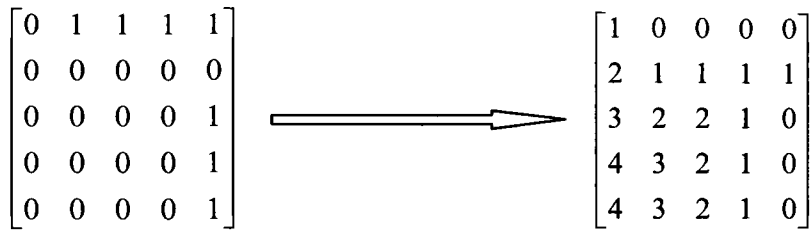


*Figure 4.6: Initial grid matrix of the previous simple map along with its distance map. In the distance map matrix, each occupied cell will have a value of 0, while free cells will have a value shows the distance (expressed in number of cells) to the nearest occupied cell.*

Distance map is an array of *Manhattan-distance* to the nearest point with value c (c is "1" in our approach) in map m for all positions of $p_1 = (x_1, y_1)$

$$d - map_c[x_1][y_1] = \min\{md(p_1, p_2) | m[p_2] = c\} \quad (1)$$

In which, $md(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$ is the *Manhattan-distance* between points $p_1$ and $p_2$. Figure 4.7 shows the initial grid map along with its equivalent distance map matrix of an area with square shape.

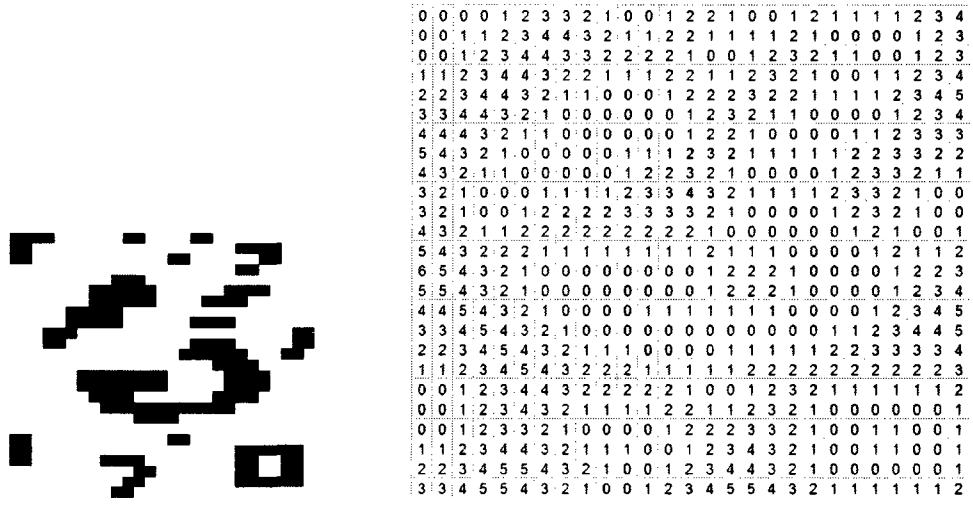


*Figure 4.7: Initial grid map of a square shape area, black cells show occupied spots while white cells are the free spaces along with its calculated distance map matrix. [7]*

Once the distance map is constructed, we will be ready to apply the candidate transformations. However, we still need a metric to measure the effectiveness of the transformations.

**Definition:** Given the distance matrices for maps $m_1$ and $m_2$, *picture distance function*, $\psi$ is declared as: [7]

$$\psi(m_1, m_2) = \sum_c d(m_1, m_2, c) + d(m_2, m_1, c)$$
$$where \qquad (2)$$
$$d(m_1, m_2, c) = \frac{\sum_{m_1[p_1]=c} \min\{md(p_1, p_2) | m_2[p_2] = c\}}{\#_c(m_1)}$$

As it can be noticed from equation (2), the idea of picture distance function is to measure the similarity between the two local maps in a mutual way in order to reduce the number of false overlaps to the lowest proportion possible. Hence, considering maps $m_1$ and $m_2$, in the first run, map $m_1$ is being kept in a stationary status while $m_2$ is being transformed to maximally overlaps $m_1$, then the distance map for the two maps is being calculated. In the second run, maps $m_1$ and $m_2$ exchange their roles and by keeping $m_2$ in a stationary status, we try to transform $m_1$ to best overlap with $m_2$. In order to get more accurate results, the values obtained from the picture distance function will be divided by the total number of occupied cells to consider the average success ratio as the final measurement criterion. It is clear that the most desired transformations will be those with picture distance values close to zero.

## 4.5 Failure detection

Although the picture distance function is aimed to find the best fitting candidate transformations, it is never guaranteed that the resulted partial maps will overlap each other in all cases and also it is possible that such an overlap does not exist even! In other words, the solution set found by the picture distance function could

still be a false positive. Therefore a mechanism is needed to verify the correctness of the proposed transformations. Fortunately by introducing a simple metric called *acceptance index* ($\omega$), it is possible to perform this verification task easily and efficiently.

**Definition:** *acceptance index,* $\omega$ is the ratio of the number of matching cells (both 0's and both 1's) in the maps being merged to the total number of cells and is defined as follows: [10]

$$\omega(M_1,M_2)=\begin{cases}0 & \text{If } sim(M_1,M_2)=0 \\ \dfrac{sim(M_1,M_2)}{sim(M_1,M_2)+dsim(M_1,M_2)} & \text{If } sim(M_1,M_2)\neq 0\end{cases} \quad (4)$$

Therefore, only values close enough to 1 for $\omega$, show a real overlap. Since these values are indiscrete, a threshold has to be defined to separate successful from unsuccessful merges. Based on different experiments conducted in the work of [7], it is indicated that values over 0.98 for $\omega$ shows a reliable threshold in order to confirm the success of the merge attempts, especially considering the fact that the best false positive gives values of $\omega$ pretty lower than 0.90. Hence, if the acceptance index satisfies the threshold, the overlap is confirmed and a new (partial) global map is being generated which will be used as the updated belief of the mapping robots (the ones whose partial maps were merged). Otherwise, the overlap is dismissed and the associated transformation will be reversed and the robots resume their own mapping task until next meet.

### Algorithmic notation

As a conclusion to this chapter, Table 4.1 summarizes all above explained steps in a form of pseudo-code. Initially there are N robots within N imaginary mapping clusters performing independent mapping task. During this individual mapping task, each robot tries to make a guess about the pose of other exploring robots within its local map. When a pair of robots are within the communication range of

each other and are able to exchange information, they will try to localize each other in their local maps using a modified version of partial localization and particle filter proposed by Howard [24]. If the mutual localization process was successful (Line 6), the distance map matrix for the local map will be created using equation (1) - explained in the previous section (Line 7). In the next step, a set of candidate transformations will be chosen using equations (2) and (3) - explained in previous section. The validity of these transformations needs to be verified and this duty is being performed using the *acceptance index* (Line 9). If the selected transformation was acceptable, the merge operation will be confirmed and a new update in the mapping belief of both robots will be adopted (Line 11). In fact, the obtained combined cluster makes up the union of the local maps of both robots right before the moment of the merge operation. On the other hand, if the evaluation process invalidates the associated transformation, it will be reversed (Line 15) and the robots continue their mapping tasks based on their pervious mapping statuses until next meet (Line18)

Algorithm 1 Clustered map merging in Multi-robot SLAM

```
1:  N robots perform SLAM (individually),      //initially N single clusters
2:  while 1 do
3:    if any two robots meet and their maps are not merged before then
4:      Each robot takes a relative distance and bearing measurement
5:      Each robot tries to localize the colleague robot in its local map using partial localization [24]
6:      if successful localization then
7:        Build the distance map matrices [(1)]
8:        Determine the candidate transformations between the two robots' maps [(2) and (3)]
9:        Verify the correctness of the transformation [(4)]
10:       if acceptable then
11:         Build the partial global map and update the beliefs of both robots
12:         Robots with merged maps continue mapping within the cluster
13:       end if
14:     else
15:         undo the last transformation
16:   end if
17:  end if
18: Robots without merged maps perform SLAM
19: end while
```

*Table 4.1: Algorithm 1, pseudo-code for the proposed method of Clustered map merging in multi-robot SLAM.*

# Chapter 5

# IMPLEMENTATION AND EXPERIMENTAL RESULTS

In order to verify the performance of our proposed method and also to test the efficiency of the algorithm, we have conducted several experiments using Microsoft Robotics Developer Studio simulation environment. In the first section of this chapter we will present the implementation details of our experiments including platform, programming environment and implementation of the code in MRDS simulation environment, while the details of each experiment and its related results will be presented in section 5.2. Section 5.3 consists of a study and analysis on the obtained results.

## 5.1 Implementation details

### 5.1.1 Why MRDS?

Microsoft® Robotics Developer Studio is a powerful application package which can be used for the design and implementation of different robotics applications. MRDS supports a wide range of robotics platforms by either running directly on that platform or controlling it from a Windows device by means of a communication channel such as Wi-Fi or Bluetooth®. [46]

An important characteristic of MRDS is the ease of full integration with .net technology development tools of Microsoft Visual Studio which provides a complete development environment for all kinds of programmers from hobbyist to professional system programmers. In fact, MRDS fully provides code portability and reusability features. Furthermore, MRDS is equipped with a strong interactive simulation environment where the output of the created projects can

be forwarded and watched on the screen which can save the costs and space needed especially during the design and test stages.

### 5.1.2 Robot

We have decided to use Pioneer 3DX robot (Figure 5.1) for our experiments. Pioneer 3DX (P3DX) from Mobile Robots Inc. is an advanced research machine that can has an on-board PC, a range of sensors and it communicates via WiFi (Wireless Ethernet). The P3DX is supported by Microsoft robotics applications in both hardware and simulation. In fact, Pioneer 3DX is a powerful mobile robot in which an accurate motion model and a precise perception system are encapsulated in one machine.

Moreover, Pioneer 3DX is a multi-purpose robot, used for research and applications involving: mapping, teleportation, localization, monitoring, reconnaissance, vision and other applications [54]. Moreover, Pioneer 3DX runs well on hard surfaces and it can traverse low sills and climb most wheelchair ramps. A summary of operations manual is provided in Appendix A.

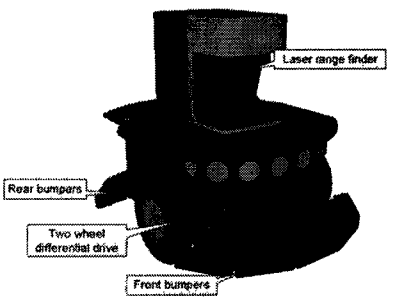


*Figure 5.1: Pioneer3DX mobile robot is a powerful multi-purpose machine equipped with accurate sensors [53]*

**Laser range finder in Pioneer 3DX**

3DX robot is equipped with SICK laser range finder sensor that can output range measurements from an angle of 100° or 180° planar field of view. It has a vertical resolution of 0.25°, 0.5° or 1.0°, demonstrating that the width of the area the laser beams

measure is 0.25°, 0.5° or 1.0° wide. The scan rate is10-12 scans/second with 4 cm range resolution while the maximum range is 50-80 meters [53]. The accuracy of the LRF will drop sharply with the existence of mirrors, glass, and matte black obstacles. In such a case, a complementary sensor, like sonar will be necessary to amend the inaccuracy of the LRF sensor. A typical laser scanner output will look like the following:

2.98, 2.99, 3.00, 3.01, 3.00, 3.49, 3.50, ..., 2.20, 8.17, 2.21

The above numbers show the range readings from right to left in terms of meters.

### 5.1.3 Mapping environment

We have used two different environments for the experiment stage. The first experimental environment is an area enclosed in a rectangular shape with dimensions of 24m x 18m (*Area1*). The second environment (*Area2*) resembles an L-shape area with dimensions of 36m x 27m. The difference between the two areas goes back to the shape and position of the walls and other obstacles. *Area1* includes a set of unorganized walls which will be used in our later performance evaluation of the proposed method representing an asymmetric environment (see Figure 5.2). On the other hand, *Area2* is an ordinary office space with several rooms and corridors with regular partitioning walls and entrance doors. The non-similar corners of the mapping area and the partitioning walls are aimed to provide a better and more realistic measurements as well as preventing the scenario from falling into symmetrical environment exploration ambiguities. (see Figure 5.3)

Regarding the occupancy grid representation of the above environment, it is important that the right grid cell size to be chosen since it plays a role in the accuracy of our results. The grid size should be determined based on the size of the robot and also the shape of the mapping environment. Since the robot diameter is almost 50cm, then a reasonable grid size is 5cm or 0.05 meters (As a rule of thumb, the grid size is one order of magnitude provided to be less than the size of the robot [55]). We can make the grid much smaller, but this will come at

the expense of increasing the computational requirements. In any case, most LRFs are only accurate to 2cm or 4cm, so grids smaller than this do not improve the measurements.

Moreover, considering the fact that these values are stored as single bytes, hence they range from 0 to 255. However, there are thresholds for occupied/vacant cells since it is not necessary for the probabilities to reach 255 or 0 respectively. Therefore the range is arbitrarily divided into three sections:

Occupied = 0 - 64

Unknown = 65 - 191

Vacant = 192 - 255

This conforms to the convention that is commonly used for occupancy grids in the literature where: Black (Occupied) represents obstacles, Grey (Unknown) is unknown, and White (Free) is free (clear or empty) space.

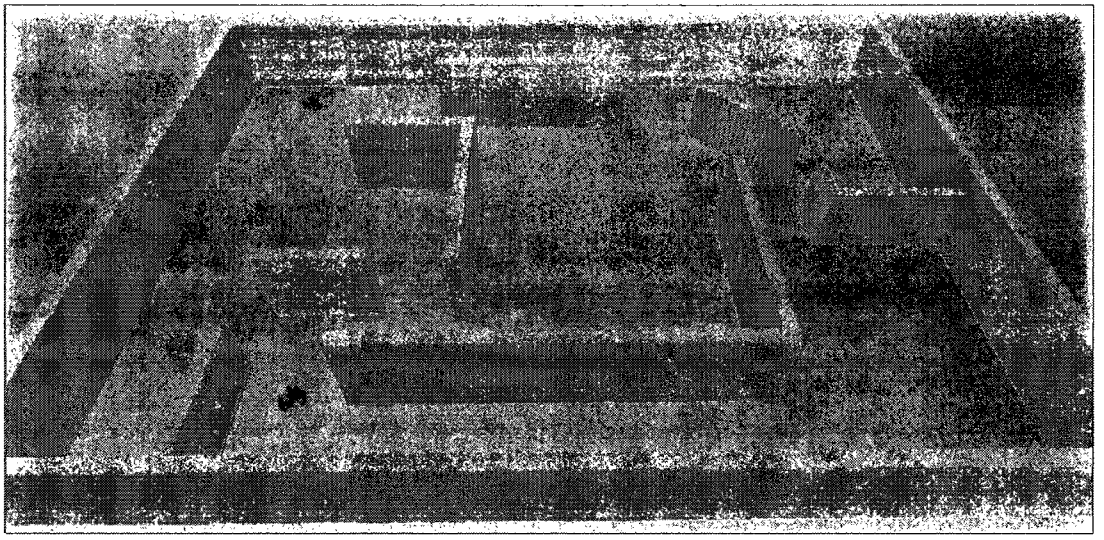

*Figure 5.2: Area1, first experimental mapping environment, an indoor office area with uneven walls used to resemble an asymmetric environment.*

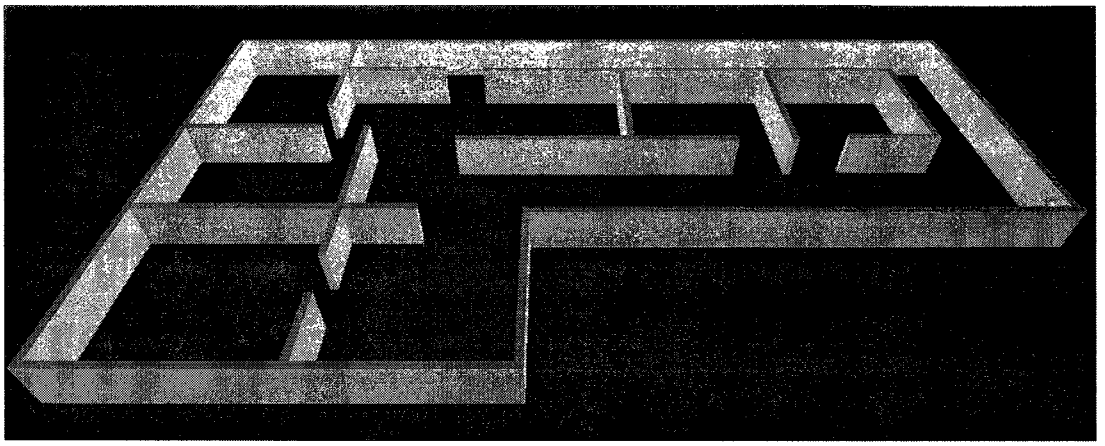

*Figure 5.3: Area2, the second experimental mapping environment, a regular indoor office area with different sized rooms and corridors.*

### 5.1.4 Programming platform

The implementation of the experiments was done using .net technology in Microsoft Visual Studio and the programming language is C#. As mentioned before, it is very convenient to direct the output of projects created in MVS development environment to MRDS simulation environment using *manifest* facility. A manifest is a special platform aimed to build the structure and shape of the output (simulation environment) and holds the required directions and configurations necessary to construct the required simulation environment. In fact it acts as a configuration file for the MRDS simulation environment.

The single robot SLAM method used in our implementation is based on Simple Mapping Utilities (pmap) SLAM implementation by A. Howard [24]. The original code was in C++ and it has been rewritten in C# to be used in our implementation. The multi-robot extension is based on the idea of grid similarities and picture distance functions proposed by [7] and [12]. Also we have used the IR sensor for the purpose of distance estimation between the mapping robots.

## 5.2 Simulation experiments

We have conducted two set of simulation experiments in *Area1* and *Area2* to test the performance of our method. In each of them, two mapping robots start their

mapping tasks from two different locations (the first one starts from the middle point while the second starts from the left part of the area). Each mapping experiment includes a three sub-experiments which are, sole mapping, co-operative dual robots mapping without our proposed enhancement and finally dual robot mapping with the in advance pairwise partial localization enhancement. The simulation experiments were performed on a laptop computer with 4GB of RAM and a Dual-core T4200 Pentium CPU @ 2 GHz.

### 5.2.1 Gaussian *pdf* parameters

As mentioned earlier, the essence of our approach is to consider and apply only an effective subset of the candidate transformations at each data exchange instance between the two robots. Our filtering mechanism is a Gaussian *pdf* which reduces the cloud of the possible transformations to those falling in an enclosed Gaussian distribution area around the point of meeting. The related Gaussian *pdf* has a mean of $\mu_c$ and covariance of $\sum_k^2$ in which, $c$ is the point of meeting and $k = \alpha.grid_{size}$ is a constant coefficient which is dependant to the grid cell size and also the mapping environment shape. Hence $k$, is the first value that needs to be determined by our experiments.

Since the grid size is 0.05 meters (5 centimeters) and the robot diameter is 0.5 m, then the maximum proper value for *k* will be 0.5 m. in other words the upper-bound for $\alpha$ will be 10. Therefore we will examine the performance of our method for the integer values of $\alpha$ ranging in the interval of $1 \leq \alpha \leq 10$. In fact when $\alpha \longrightarrow 0$, the set of candidate transformations $\longrightarrow \Phi$ (empty set) and hence, the number of data exchange operations $\longrightarrow$ 0 too and practically the multi-robot mapping task becomes a single robot mapping task in which the other robot becomes an obstacle to be prevented.

To evaluate the influence of $\alpha$ on the mapping process, we define the following metrics:

$$Transformation\ Effectiveness = \frac{Number\ of\ successful\ transformations}{Number\ of\ data\ exchanges} \quad (1)$$

(2)

$$Transformation\ Ratio = \frac{Number\ of\ successful\ transformations}{Total\ number\ of\ successful\ transfor\ mations\ for\ all\ values}$$

(3)

$$Weighted\ Throughput = \text{Transformation Effectiveness} * \ Transformation\ Ratio$$

In equation (1), we consider the ratio of successful transformations to the total number of transformations performed in a mapping task. This metric is a suitable criterion to measure the performance of the mapping methods. On the other hand, equation (2) calculates the *Transformation ratio* for a multi-fold mapping experiments set and finds the weight (significance) of each fold. Finally, in equation (3) we calculate the *weighted throughput* for each mapping task trial.

In order to find the best value of $\alpha$ for our mapping method implementation, we have conducted three experiments for each value of $\alpha$ ranging from 1 to 10 to find the related average weighted throughput. These experiments were performed based on our proposed multi-robot mapping method in *Area2* using two mapping robots. The results obtained from these experiments are shown in Table 5.1. We can observe from these results that the best average weighted throughput of 0.176 was achieved by $\alpha$= 7. This result means that 17.6% of the

| | Weighted Throughput | | | |
|---|---|---|---|---|
| Run | First | Second | Third | Average |
| $\alpha$ =1 | 0.007 | 0.011 | 0.009 | 0.009 |
| $\alpha$ =2 | 0.039 | 0.043 | 0.031 | 0.038 |
| $\alpha$ =3 | 0.042 | 0.048 | 0.052 | 0.047 |
| $\alpha$ =4 | 0.101 | 0.092 | 0.084 | 0.092 |
| $\alpha$ =5 | 0.132 | 0.149 | 0.118 | 0.133 |
| $\alpha$ =6 | 0.149 | 0.148 | 0.162 | 0.153 |
| $\alpha$ =7 | 0.158 | 0.187 | 0.184 | **0.176** |
| $\alpha$ =8 | 0.162 | 0.169 | 0.155 | 0.162 |
| $\alpha$ =9 | 0.121 | 0.108 | 0.109 | 0.113 |
| $\alpha$ =10 | 0.082 | 0.081 | 0.068 | 0.077 |
| Total | | | | 1.000 |

*Table 5.1: average weighted throughput for each value of $\alpha$ ranging from 1 to 10. On $\alpha$ = 7* we obtained the best weighted throughput

successful transformations made by all values of $1 \leq \alpha \leq 10$ were made in the fold of $\alpha$= 7. A graph representing the relationship between the values of $\alpha$ and the weighted throughput is provided in Figure 5.4.

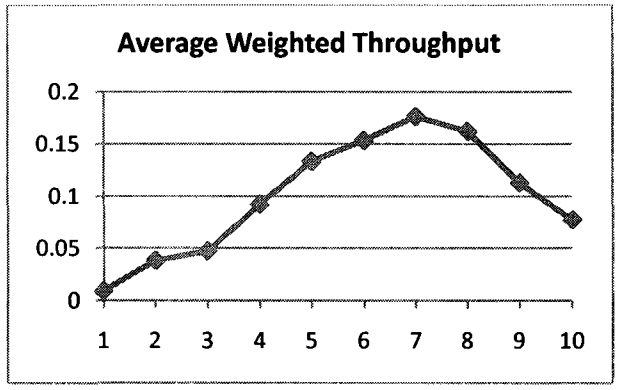


*Figure 5.4: Average Throughput for different values of $\alpha$*

**Discussion:**

To analyze the trend of the above graph, it has to be mentioned that initially when $\alpha$ has the value of 1, the Gaussian *pdf* would have very small values around the mean (the point of meeting coordinates) and hence, many candidate transformations (both successful and unsuccessful) will be dismissed as they lay outside of the consideration region. In other words the multi-robot method will have a behavior similar to single robot mapping method. As the value of $\alpha$ grows the radius of the consideration region extends too and covers a wider set of candidate transformations. In some points of this range, our algorithm will show its best performance. On the other hand, when $\alpha$ continues in taking higher values, the region of consideration becomes huge and our algorithm starts to lose its efficiency since the number of false positive transformations inside the region of consideration will increase rapidly and our method will behave similar to the Basic multi-robot mapping approach.

As a result of the above experiments and discussion, a value of 7 will be assigned to the constant $\alpha$ throughout the implementation of our proposed multi-robot mapping algorithm for all next mapping experiments.

### 5.2.2 Mapping experiments

#### 5.2.2.1 Area1 Mapping experiments

In the first set of these experiments, *Robot1* and *Robot2* start to map *Area1* individually. At each run the other robot has been eliminated to prevent the mapping robot from dealing with an unintended obstacle. Our target was to have 10 successful experiments with complete coverage of the mapping area. To accomplish this aim, we had to repeat the mapping experiment 14 times. The other 4 unsuccessful experiments failed to completely cover the mapping area after 120 minutes (two hours) because the mapping robot was stuck in some parts of the map. Results for mapping time using single robot (*Robot1* and *Robot2*) are reflected in Table 5.2. Since we assume that the initial pose of the mapping robot is unknown, we treat the results obtained for both robots equally in computing the average time needed to map *Area1,* and therefore this average will be considered in the analysis and study stage.

| Area1 | Mapping Time (Minutes) | | |
|---|---|---|---|
| **Run** | **Robot1** | **Robot2** | **Average** |
| **1** | 19.18 | 18.38 | **18.78** |
| **2** | 14.54 | 14.46 | **14.5** |
| **3** | 18.00 | 17.59 | **17.8** |
| **4** | 14.38 | 12.80 | **13.59** |
| **5** | 15.59 | 19.27 | **17.43** |
| **6** | 12.35 | 14.85 | **13.6** |
| **7** | 17.02 | 21.20 | **19.11** |
| **8** | 11.85 | 16.53 | **14.19** |
| **9** | 13.40 | 15.20 | **14.3** |
| **10** | 15.55 | 18.76 | **17.16** |
| **Average** | **15.19** | **16.90** | **16.05** |

*Table 5.2: Area1 mapping time for single Robot1 and Robot2 individually*

Table 5.3, on the other hand shows the mapping time for *Area1* using the Basic multi-robot SLAM (without the proposed enhancement of in advance

pairwise partial localization). We managed to get 10 successful fully covered maps after repeating the experiment for 12 times.

| Experiment run | Number of data exchange | Number of successful transformations | Collective mapping time (Minutes) |
|---|---|---|---|
| **1** | 16 | 9 | **14.78** |
| **2** | 19 | 12 | **13.58** |
| **3** | 18 | 11 | **12.80** |
| **4** | 10 | 6 | **12.59** |
| **5** | 11 | 3 | **16.65** |
| **6** | 18 | 14 | **12.10** |
| **7** | 22 | 11 | **15.91** |
| **8** | 17 | 10 | **13.78** |
| **9** | 18 | 12 | **12.32** |
| **10** | 22 | 17 | **11.05** |
| **Average** | **17.1** | **10.5** | **13.56** |

*Table 5.3: Area1 mapping statistics using basic multi-robot SLAM implementation performed by Robot 1 and Robot2*

The next step was to implement our proposed enhancement to the Basic multi-robot mapping and map merging technique. The results are reflected in Table 5.4.

| Experiment run | Number of data exchange operations | Number of successful transformations | Collective mapping time (Minutes) |
|---|---|---|---|
| **1** | 9 | 7 | **10.11** |
| **2** | 8 | 6 | **11.23** |
| **3** | 7 | 7 | **10.02** |
| **4** | 7 | 6 | **13.27** |
| **5** | 4 | 4 | **9.88** |
| **6** | 9 | 7 | **11.20** |
| **7** | 10 | 8 | **13.05** |
| **8** | 8 | 7 | **11.82** |
| **9** | 8 | 6 | **14.02** |
| **10** | 4 | 4 | **9.85** |
| **Average** | **7.4** | **6.2** | **11.45** |

*Table 5.4: Area1 mapping statistics using our proposed Multi-robot SLAM method with the enhancement of in-advance pair-wise partial localization*

**Observations and Case study:**

It is observable from the above statistics that our technique surpasses the performance of the Basic multi-robot SLAM method in which the average number of data exchange operations was reduced from 17.1 to 7.4 (57% decrease) while the average number of successful transformations was decreased from 10.5 to 6.2 (41% decrease) only. To better understand these results and compare the performance of the two methods, we consider the Transformation effectiveness metric of both methods. Based on the results obtained from the above experiments, we have:

Effectiveness for Basic multi-robot SLAM: $10.5 / 17.1 \approx 0.61$

Effectiveness for our proposed Multi-robot SLAM: $6.2 / 7.4 \approx 0.84$

Hence, the transformation effectiveness shows an improvement of 23 points from 61% to 84% (almost 38% increase). In other words, the rate of average false positive transformations went down from 39% (1-0.61) to 16% (1-0.84). Furthermore, the average mapping time shows an improvement of 19% from 13.65 to 11.45 minutes.

Figure 5.5 demonstrates an example of a successful map merging task in which the partial map drawn by *Robot2* (shape a) was transformed to best overlap with the partial map of Robot1 (shape b) and then they were fused together to build the new mapping belief for both robots (shape c) right after the moment of the merge process. On the other hand, Figure 5.6 shows an example of unsuccessful transformation attempt during the mapping process of *Area1* in which the local map drawn by *Robot2* (shape a) was flipped vertically (half plane rotated) and wrongly positioned on the upper part of local map drawn by *Robot1* (shape b) and created a partial global map (shape c). This situation was due to the fact that the heuristic search function had given a positive transformation sign to proceed with that specific transformation although it was a wrongly picked one. Finally the acceptance index managed to catch the fault made, and the wrong transformation was reversed.

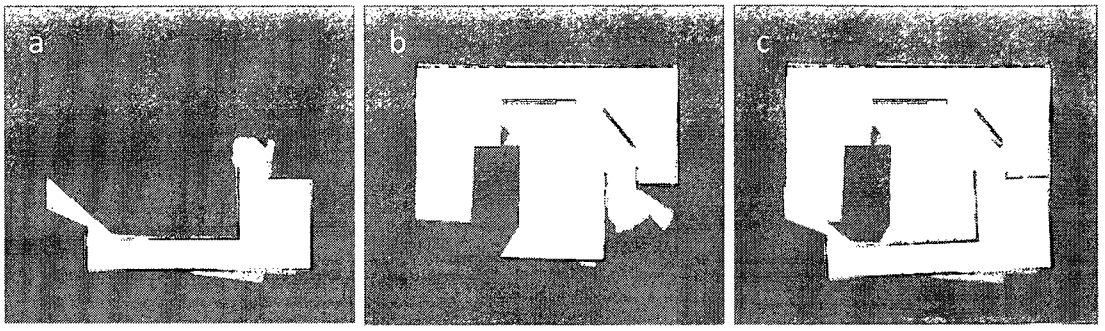


*Figure 5.5: an example of successful transformation and hence merging procedure in Area1, a is the partial map drawn by Robot2, b is the partial map drawn by Robot1 and c is the partial global map right after the merge process.*

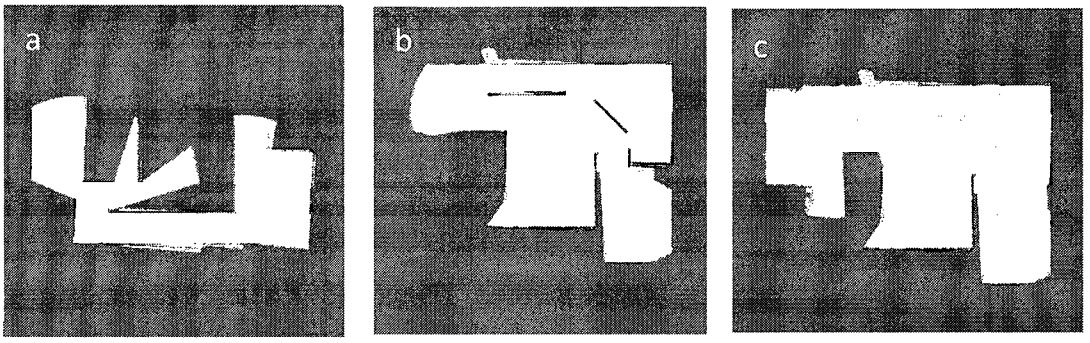


*Figure 5.6: an example of unsuccessful transformation in Area1 multi-robot mapping procedure. Local map drawn by Robot2 (shape a) was flipped vertically (half plane rotated) and wrongly positioned on the upper part of local map drawn by Robot1 (shape b) and created a partial global map (shape c). The distance map heuristic function had given a positive transformation sign while the acceptance index finally caught the fault made the wrong transformation.*

### 5.2.2.2 Area2 mapping experiments

In the first set of these experiments, *Robot1* and *Robot2* start to map *Area2* individually. At each run the other robot has been eliminated to prevent the mapping robot from unintended obstacle. To get the results for 10 successful runs, we had to repeat the mapping experiment 15 times. Results for mapping time using single robot (*Robot1* and *Robot2*) are reflected in Table 5.5.

The next step was to redo the experiment using the basic multi-robot method. Table 5.6 shows the results obtained from the collective mapping task

| | Time (Minutes) | | |
|---|---|---|---|
| RUN | Robot1 | Robot2 | Average |
| 1 | 41.58 | 29.78 | **35.68** |
| 2 | 29.08 | 33.93 | **31.51** |
| 3 | 24.45 | 50.67 | **37.56** |
| 4 | 53.1 | 43.25 | **48.18** |
| 5 | 38.15 | 35.23 | **36.69** |
| 6 | 23.58 | 36.29 | **29.94** |
| 7 | 33.9 | 27.89 | **30.9** |
| 8 | 26.18 | 34.3 | **30.24** |
| 9 | 37.1 | 37.2 | **37.15** |
| 10 | 26.7 | 23.93 | **25.32** |
| **Average** | **33.38** | **35.25** | **34.31** |

*Table 5.5: Area2 mapping time for single Robot1 and Robot2*

performed by Robot1 and Robot2 to build the global map of *Area2*.

Finally the experiment was repeated to engage our proposed enhancement to the Basic multi-robot mapping and map merging technique and verify its functionality and performance in environments similar to *Area2*. The results are reflected in Table 5.7.

| Experiment run | Number of data exchange operations | Number of successful transformations | Collective mapping time (Minutes) |
|---|---|---|---|
| 1 | 30 | 21 | **25.45** |
| 2 | 27 | 20 | **23.18** |
| 3 | 25 | 18 | **22.80** |
| 4 | 21 | 17 | **19.90** |
| 5 | 22 | 12 | **26.65** |
| 6 | 21 | 9 | **28.10** |
| 7 | 26 | 15 | **25.91** |
| 8 | 30 | 25 | **21.75** |
| 9 | 25 | 20 | **24.12** |
| 10 | 22 | 17 | **21.15** |
| **Average** | **24.9** | **17.4** | **23.90** |

*Table 5.6: Area2 mapping statistics using basic Multi-robot SLAM implementation performed by Robot1 and Robot2*

| Experiment run | Number of data exchange | Number of successful transformations | Collective mapping time (Minutes) |
|---|---|---|---|
| 1 | 19 | 17 | **23.40** |
| 2 | 17 | 16 | **21.08** |
| 3 | 16 | 14 | **20.50** |
| 4 | 22 | 19 | **19.80** |
| 5 | 17 | 15 | **21.55** |
| 6 | 14 | 11 | **22.30** |
| 7 | 11 | 10 | **23.82** |
| 8 | 20 | 17 | **19.75** |
| 9 | 16 | 14 | **21.32** |
| 10 | 13 | 9 | **20.11** |
| **Average** | **16.5** | **14.2** | **21.36** |

*Table 5.7: Area2 mapping statistics using our proposed Multi-robot SLAM method with the enhancement of in-advance pair-wise partial localization*

**Observations and Case study:**

It is observable from the obtained statistics that using our approach to build the map for *Area2* has surpassed the performance of the Basic multi-robot SLAM in which the average number of data exchange processes was reduced from 24.9 to 16.5 (34% decrease) while the average number of successful transformations was decreased from 17.4 to 14.2 (18% decrease only). Therefore to compare the overall performance of both algorithms, we use the transformation effectiveness metric again for *Area2*, hence based on the results found from the above experiments, we have:

Effectiveness for the Basic Multi-robot SLAM: $17.4 / 24.9 \approx 0.71$
Effectiveness for our proposed Multi-robot SLAM: $14.2 / 16.5 \approx 0.86$

It can be noticed that the transformation effectiveness shows an improvement of 15 points from 71% to 86% (almost 21% increase). In other words, the rate of average false positive transformations went down from 29% (1-0.71) to 14% (1-0.86). Furthermore, the average mapping time shows an improvement of almost 12% from 23.90 to 21.36 minutes.

Figure 5.7 demonstrates an example of a successful map merging task in which the partial map drawn by *Robot2* (shape a) was transformed to best overlap with the partial map of Robot1 (shape b) and they were fused together to build the new mapping belief for both robots (shape c) right after the moment of the merge process. Figure 5.8 demonstrates the final global map for Area2 drawn by Robot1 and Robot2 using our proposed enhancement for multi-robot SLAM. This final map was a result of multiple stages of transformations and local map merging tasks.

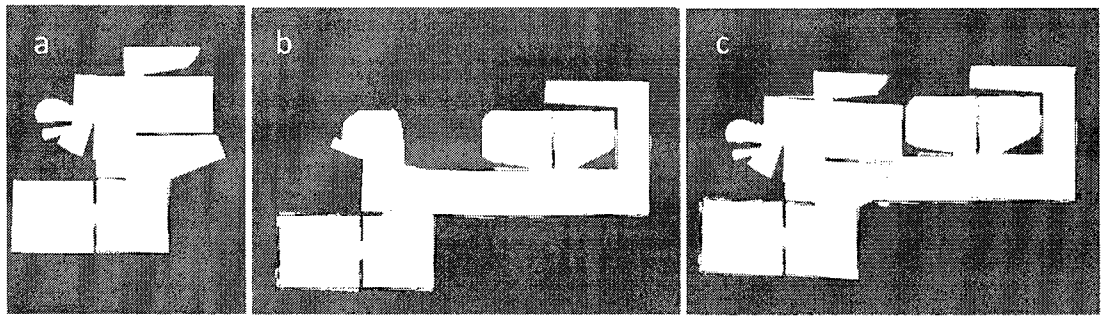


*Figure 5.7: an example of successful transformation and hence merging procedure in Area2, a is the partial map drawn by Robot2, b is the partial map drawn by Robot1 and c is the partial global map right after the merge process.*

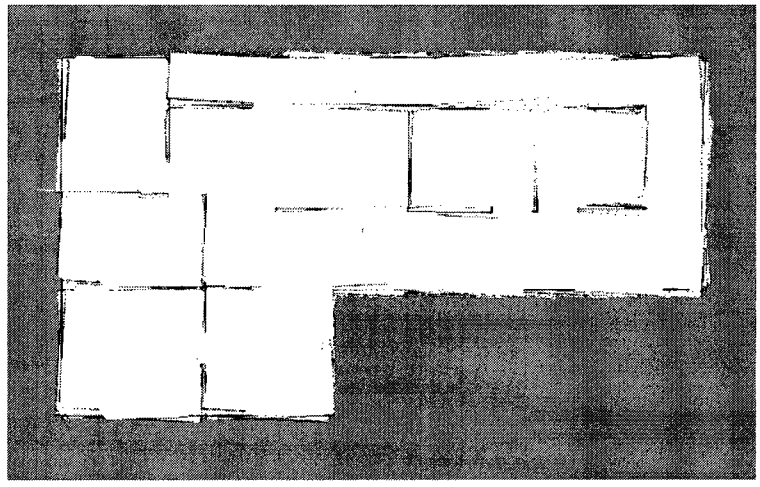

*Figure 5.8: Area2 final map drawn by Robot1 and Robot2 after several stages of transformations and local map merging tasks.*

## 5.3 Analysis and discussion

The main aim of our approach was to challenge the feasibility of having a kind of refinement procedure over the possible candidate transformations by eliminating those false positives before physical occurrences versus having a larger set of candidate transformations and reversing those do not comply with our pickup metrics in a later process.

Considering the transformation effectiveness ratios and also the average mapping time of multi-robot SLAM with and without our proposed method, there is a clear evidence that reversing undesired transformations has a higher cost for the system rather than the proposed transformations filtering mechanism. The results obtained through our experiments confirm the fact that the excess time spent in our proposed pairwise partial localization approach not only did not increase the overall mapping time, but also it contributed effectively in reducing it. This achievement is due to the reduced number of false positives in map transformations and consequently eliminating the time needed to reverse them, achieved by our proposed method.

**Mapping environment shape impact:**

**Observation:**

Beside the fact that *Area1* has simpler map structure than *Area2,* it was noticed that *Area2* has a lower number of false positive transformations for both multi-robot SLAM methods (1- 0.71= 29% for the Basic algorithm SLAM and 1- 0.86 = 14% for the proposed method versus 1-0.61 = 39% for the Basic SLAM and 1-0.84 = 16% for our proposed method) which infers the fact that areas with more distinct regions (dissimilar rooms and corridors in our example) have a higher possibility of being detected and identified by mapping robots and hence a higher rate of successful transformations. The enhancement of our proposed method for this area was 21% (from 71% to 86%), while for areas with less distinct regions

like *Area1*, the rate of improvement achieved by our method was 38% (from 61% to 84%). (See Figure 5.9)

Also based on the obtained results, it is evident that our proposed method has almost the same performance rate in both areas (84% for Area1 and 86% for Area2). This observation implies that there is a high degree of independence relation between the shape of the mapping area and the rate of successful transformations in our approach. On the other hand, the Basic multi-robot SLAM performs differently in the two environments (61% for *Area1* and 71% for *Area2*) and therefore it can be deduced that it is dependant to the shape of the mapping area and hence the transformation effectiveness could differ in a wide interval range. (see Figure 5.10)

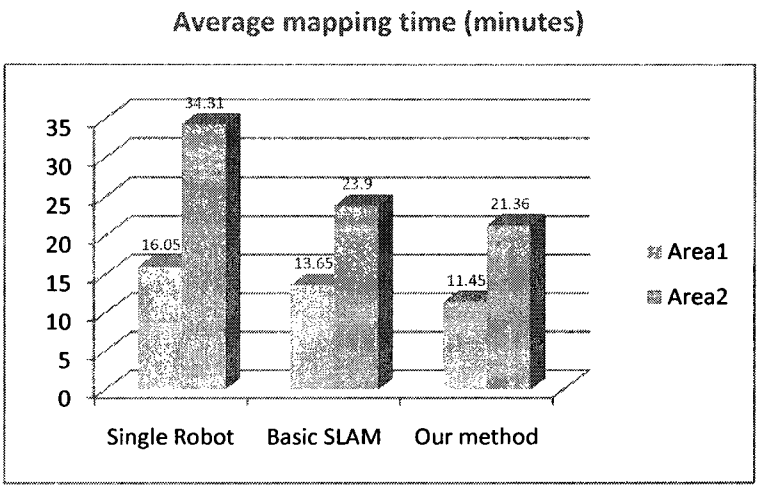


*Figure 5.9: overall mapping time for Area1 and Area2 using single mapping robot, Multi-robot Basic SLAM and our proposed Multi-robot method. It is noticeable that our proposed method performed better in both environments in the sense of the overall mapping time*

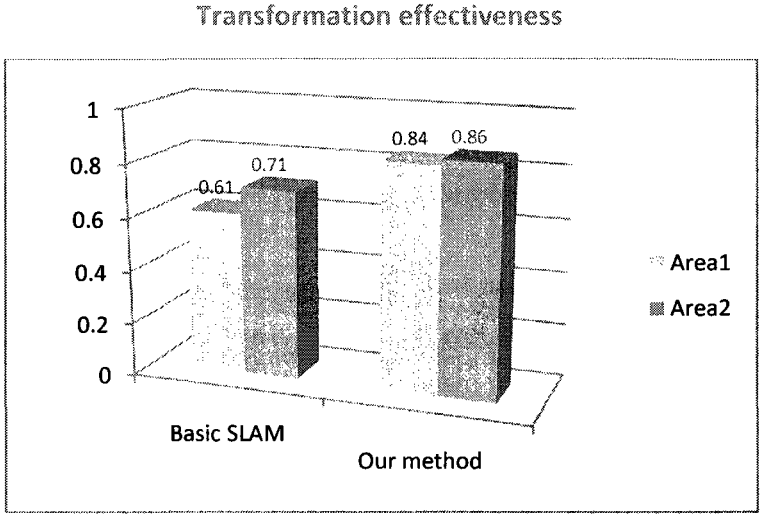


*Figure 5.10: Transformation effectiveness for different mapping methods. It can be seen that the Basic SLAM performs differently in the two areas while our proposed methods has almost the same rate of transformation effectiveness and therefore it is independent from the environment shape*

# Chapter 6

# CONCLUSION AND FUTURE WORK

## 6.1 Conclusion

Our proposed method to improve the process of map merging in multi-robot SLAM can be added as an enhancement module to some of the current multi-robot mapping methods which are based on gird similarities and transformations. In particular, the proposed approach enables a team of robots to explore the environment more efficiently from unknown locations.

In particular, each robot is located in an imaginary cluster holding its local map, explores the environment independently until it can communicate with another robot. At this point, each robot will try to localize the other robot into it's local map. In the case of success, a subset of candidate transformations satisfying certain conditions will be considered for overlap evaluation process. As a result, those picked up transformations will be applied to the common portions of the source local map to best overlap with the target one and finally fuse both local maps together to build a partial global map of the environment.

During the operation of map merging, a new mapping cluster will be created by combining the initial clusters. In fact, all robots within the new cluster will share the map and pose beliefs right after the moment of the merging operation. The ultimate goal will be a single cluster which covers the entire area of the environment being mapped and clearly includes all the members of the team of robots.

In order to test the efficiency and performance of our proposed method, we have tested the implementation of our technique using the simulation environment of MRDS with two different mapping environments. A noticeable improvement in the overall mapping time as well as the percentage of successful transformations for both environments was achieved.

Also as a generalization to this method to include the case of having more than two robots in the field, the proposed algorithm and concept are still applicable since they provide a method to integrate the local maps constructed by more than one robot and build a united global view of the environment. In fact, the essence of our proposed enhancements (both the pairwise partial localization and the Gaussian distribution around the point of meeting between the two robots) are based on mutual relations between any given pair of robots set to merge their local maps. Therefore, considering the fact that the history of the map merging process does not play a role in the overall procedure and also that a merged map resulted from a merge process between two local maps still can be considered as a local map, a merge process between more than two robots can be done using the same concept of merging two robot and by a set of non significant changes in the implementation part.

In the scenario of having more than two local maps eligible for a merge process at the same time, a sequential method can be considered in which the order the local maps are being picked up could be in a random way. A more in depth discussion about the generalization of our proposed method is left for future work.

## 6.2 Limitations of the proposed method

Two main limitations affect the functionality of the proposed algorithm. First, it is required to assume that the maps being merged have been built using the same scale. The algorithm is incapable of determining whether or not the two maps being merged need to be magnified in order to match with each other before the transformations. Second, in order to have a successful merge task, it is necessary that the two maps being merged exhibit a certain degree of overlapping. If this condition cannot be satisfied, the proposed method is unlikely to find the appropriate transformation, although the acceptance index will indicate that the candidate transformation and consequent merging operation should be

discarded. It should however be noted that most map merging methods share these limitations as well.

## 6.3 Future work

Although we have implemented our proposed enhancement to a specific multi-robot mapping algorithm, it would appear that the idea of a generalization of this technique to include other grid similarities based multi-robot mapping methods could be beneficial in order to increase their overall mapping performance. The current implementation does not serve this goal since it is part of the mapping code kernel with tens of direct dependencies with other parts of the code. Therefore, to be able to achieve this goal, it would be necessary to review the structure of the current implementation and build a system with a minimum set of dependencies between this module and the mother code of the multi-robot mapping implementation.

Also as a possible improvement to our method, it would appear that building a supportive more in depth mathematical model for this approach and testing the impact of the different parameters of that model could lead to a more robust technique with higher performance which can achieve even better results than those obtained based on a pure experimental analysis.

# REFRENCES


[1] ADLURU, N., LATECKI, LJ, SOBEL., M, AND LAKAEMPER, R. 2008. Merging maps of multiple robots, *ICPR 2008. 19th International Conference on Pattern Recognition,* 1 – 4

[2] AMIGONI, F., GASPARINI, S., AND GINI, M. 2006. Building Segment-Based Maps Without Pose Information, *Proceedings-IEEE*, 94(7):1340- 1359

[3] AMIGONI, F., GASPARINI, S., GINI, M. 2004. Scan matching without odometry information, *In Proceedings of First International Conference on Informatics in Control, Automation and Robotics,* 349-352

[4] AMIGONI, F., GASPARINI, S., GINI, M. 2005. Merging Partial Maps Without Using Odometry, *Multi-robot Systems. From Swarms to Intelligent Automata* Volume III, *Springer Netherlands,* 133-144

[5] BALAGUER, B., CARPIN, S. 2008. UC Mercenary Team Description Paper: Robocop 2008 Virtual Robot Rescue Simulation League, *School of Engineering, University of California, Merced*

[6] BALAKIRSKY, S., CARPIN, S., KLEINER, A., LEWIS, M., VISSER, A., WANG, J. and AMOS ZIPARO, V. 2007. Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from RoboCup rescue, *Journal of Field Robotics,* 24(11): 943–967

[7] BIRK, A. and CARPIN, S. 2006. Merging occupancy grid maps from multiple robots, *Proceedings-IEEE,* 94(7):1384 – 1397

[8] BIRK, A. and CARPIN, S. 2006. Rescue robotics-a crucial milestone on the road to autonomous systems, *Advanced Robotics,* 20(5):595-605

[9] BURGARD, W. MOORS, M. STACHNISS, C. SCHNEIDER, F.E. 2005. Coordinated multi-robot exploration, *IEEE Transactions on Robotics*, 21(3): 376- 386

[10] CARPIN, S. 2008. Fast and accurate map merging for multi-robot systems, *Autonomous Robots, Springer,* 25(3): 305-316

[11] CARPIN, S. 2008. Merging Maps via Hough Transform, *IEEE/RSJ International conference on Intelligent Robots and Systems (IROS),* 1878 – 1883

[12] CARPIN, S., BIRK, A., JUCIKAS, V. 2005. On map merging, *Robotics and autonomous systems, Elsevier* 53(1):1-14

[13] CHOSET, H., LYNCH, K. M., HUTCHINSON, S., KANTOR, G., BURGARD, W., KAVRAKI, L. E., AND THRUN, S., 2004. The wide world of mapping, *WTEC Workshop: Review of US Research in Robotics, URL: wtec.org*

[14] FARRINGTON, N., NGUYEN, H., PEZESHKIAN, N. 2005. Intelligent behaviours for a convoy of indoor mobile robots operating in unknown environments, *Proceedings of SPIE-the International Society for Optical, Mobile robots XVII,* 5609, 317–323

[15] FENWICK, J.W., NEWMAN, P.M., LEONARD, J.J. 2002. Cooperative concurrent mapping and localization, *Proceedings. IEEE International Conference on Robotics and Automation ICRA '02.* 2, 1810 - 1817

[16] FERRANTI, E., TRIGONI, N., and LEVENE, M. 2009. Rapid exploration of unknown areas through dynamic deployment of mobile and stationary sensor nodes. *Autonomous Agents and Multi-Agent Systems, Springer Netherlands*, 19(2):210-243

[17] FOX, D., KO, J., KONOLIGE, K., and STEWART, B. 2005. A Hierarchical Bayesian Approach to the Revisiting Problem in Mobile Robot Map Building, *Springer Tracts in Advanced Robotics, Springer Berlin / Heidelberg*, 15, 60-69

[18] FOX, D., KO, J., KONOLIGE, K., LIMKETKAI, B., SCHULZ, D. and STEWART, B. 2006. Distributed multi-robot exploration and mapping, *Proceedings-IEEE*, 94(7):1325-1339

[19] FUJISHIMA, H., RANKIN, E. S., GOSSAGE, M. P., CHNG, R. C., PENG NEW, AI. 2008. Multi-robot guided autonomy for indoor exploration, *26th Army science conference*, URL: asc2008.com

[20] GIANNETTI, L.; VALIGI, P. 2006. Collaboration among Members of a Team: a Heuristic Strategy for Multi-robot Exploration, *14th Mediterranean Conference on Control and Automation, MED '06*, 1 – 6

[21] HO, K. L., and NEWMAN, P. 2007. Detecting Loop Closure with Scene Sequences, *International Journal of Computer Vision, Springer*, 74(3):261-268

[22] HO, K. L., NEWMAN, P. 2005. Multiple map intersection detection using visual appearance, Conference on Computational Intelligence, *Robotics and autonomous systems, Elsevier,* 12, 114-116

[23] HOWARD, A. SUKHATME, G.S. MATARIC, M.J. 2006. Multi-robot Simultaneous Localization and Mapping Using Manifold Representations, *Proceedings of the IEEE*, 94(7): 1360-1369

[24] HOWARD, A. 2006. Multi-robot simultaneous localization and mapping using particle filters, *The International Journal of Robotics Research*, 25(12):1243-1256

[25] HUANG, W. H., BEEVERS, K. R. 2005. Topological Mapping with Sensing-Limited Robots, *Springer Berlin / Heidelberg*, 17, 235-250

[26] KOBAYASHI, F.; SAKAI, S.; KOJIMA, F. 2002. Sharing of exploring information using belief measure for multi robot exploration, *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, 2002. FUZZ-IEEE'02*, 2,1544 – 1549

[27] KOBAYASHI, F.; SAKAI, S.; KOJIMA, F. 2003. Determination of exploration target based on belief measure in multi-robot exploration, *Proceedings .IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 3, 1545 – 1550

[28] KONOLIGE, K., FOX, D., LIMKETKAI, B., KO, J., STEWART, B. 2003. Map Merging for Distributed Robot Navigation, *Proceedings IEEE/RSJ International conference on Intelligent Robots and Systems (IROS),* 1 : 212 – 217

[29] KONOLIGE, K., GUTMANN, J.S., LIMKETKAI, B. 2003. Distributed map-making, *Workshop on Reasoning with Uncertainty in Robotics, Eighteenth international joint conference on artificial intelligence*, San Francisco, CA, 1157–1164

[30] KOPERSKI, C. 2007. Multi-robot FASTSLAM for large domains, Master Thesis, *AIR FORCE INSTITUTE OF TECHNOLOGY, Wright-Patterson Air Force Base, Ohio,* URL: dtic.mil

[31] L´OPEZ, J. 2009. Workshop of Physical Agents 2009, *Journal of physical agents,* 3(1)

[32] LAKAEMPER, R., ADLURU, N., LATECKI, L J., MADHAVAN, R., 2007. Multi robot mapping using force field simulation, *Journal of Field Robotics,* 24(8&9): 747-762

[33] LAKAEMPER, R., LATECKI, J. and WOLTER, D. 2005. Incremental multi-robot mapping, *International conference on Intelligent Robots and Systems (IROS),*3846 - 3851

[34] LAKAEMPER, R., LATECKI, L.J., SUN, X. and WOLTER, D. 2005. Geometric Robot Mapping, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 3429, 11-22

[35] LEÓN, A., BAREA, R., BERGASA, L.M., LÓPEZ, E., OCAÑA, M., and SCHLEICHER, D. 2009. SLAM and Map Merging, *Journal of Physical Agents,* 3(1):13-24

[36] MARCHETTI, L. 2007. Multi-Agent Distributed Estimation, *CiteSeerX,* URL: http://www.dis.uniroma1.it/~dottoratoii/db/relazioni/relaz_lmarchetti_2.pdf

[37] MARCO, M., GARULLI, A., GIANNITRAPANI, A., VICINO, A. 2003. Simultaneous Localization and Map Building for a Team of Cooperating Robots: A Set Membership Approach, *IEEE Transactions on Robotics and Automation, 19(2):238-249*

[38] MARTINELLI, A. 2007. Improving the precision on multi robot localization by using a series of filters hierarchically distributed, *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and System,*1053 - 1058

[39] PAN., W., CAI, Z., LIU, L., CHEN, B. 2008. An Approach to Cooperative Multi-robot Map Building in Complex Environments, *The 9th International Conference for Young Computer Scientists, ICYCS 2008,*1733 - 1737

[40] PANZIERI, S., PASCUCCI, F., SANTINELLI, I., ULIVI, G. 2004. Merging topological data into KALMAN based SLAM, *Proceedings, World Automation Congress,* 15 :57 – 62

[41] PFINGSTHORN, M., SLAMET, B. and VISSER, A. 2008. A scalable hybrid multi-robot slam method for highly detailed maps, *Lecture Notes in Computer Science, Springer,* 5001, 457-464

[43] SAITOV, D., UMIROV, U., PARK, JI., CHOI, JW., and LEE, SG. 2008. Effective Map Building Using a Wave Algorithm in a Multi-robot System, *International Journal of Precision Engineering and Manufacturing,* 9(2):69-74

[43] SIM, R., LITTLE, JJ. 2006. Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters, *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2082 - 2089

[44] STEWART, B., KO, J., FOX, D., KONOLIGE, K. 2003. The revisiting problem in mobile robot map building: A hierarchical Bayesian approach, *Springer Tracts in Advanced Robotics*, 15, 60-69

[45] TANAKA, K., KONDO, E. 2008. A Scalable Localization Algorithm for High Dimensional Features and Multi Robot Systems, *IEEE International Conference on Networking, Sensing and Control, ICNSC 2008*,920 – 925

[46] TAYLOR, T. and JOHNS, K. 2008. Professional Microsoft Robotics Developer Studio, text book, ISBN: 978-0-470-14107-6

[47] THRUN, S., BURGARD, W. and FOX, D 2006. Probabilistic robotics, textbook ISBN: - 978-0-262-20162-9

[48] Wang, Z., Huang, S., Dissanayake, G. 2007. Multi-robot simultaneous localization and mapping using D-SLAM framework, *3rd International Conference on Intelligent Sensors, Sensor Networks and Information,* 317-322

[49] WILLIAMS, B., CUMMINS, M., NEIRA, J., NEWMAN, P., REID, I., TARD'OS, J. 2008. An image–to–map loop closing method for monocular SLAM, *IEEE/RSJ International conference on Intelligent Robots and Systems (IROS),*2053 – 2059

[50] WOLTER, D., LATECKI, LJ, LAKAMPER, R. An Architecture for Shape-Based Robot Mapping, URL: cs.waikato.ac.nz

[51] ZHOU, X. S., ROUMELIOTIS S. I. 2006. Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case, *Proceedings IEEE/RSJ International conference on Intelligent Robots and Systems (IROS),* 1:1785 – 1792

[52] ZIPARO, V., KLEINER, A., NEBEL, B. and NARDI, D. 2007. Rfid-based exploration for large robot teams, *Proceedings of the IEEE International conference on Robotics and Automation, cs.virginia.edu*, 4606-4613

[53] URL: www.wiphala.net

[54] URL: http://www.ist.tugraz.at/_attach/Publish/Kmr06/pioneer-robot.pdf

[55] URL: http://www-robotics.usc.edu/~ahoward/pmap/index.html

# APPENDIX A: PIONEER 3DX ROBOT

Pioneer 3DX was first supplied in the summer of 2003. It uses a microcontroller based on the Hitachi H8S microprocessor, with new control systems software (AROS) and I/O expansion capabilities. The Pioneer 3 robot also had new, more powerful motor/power system for better navigational control and payload.

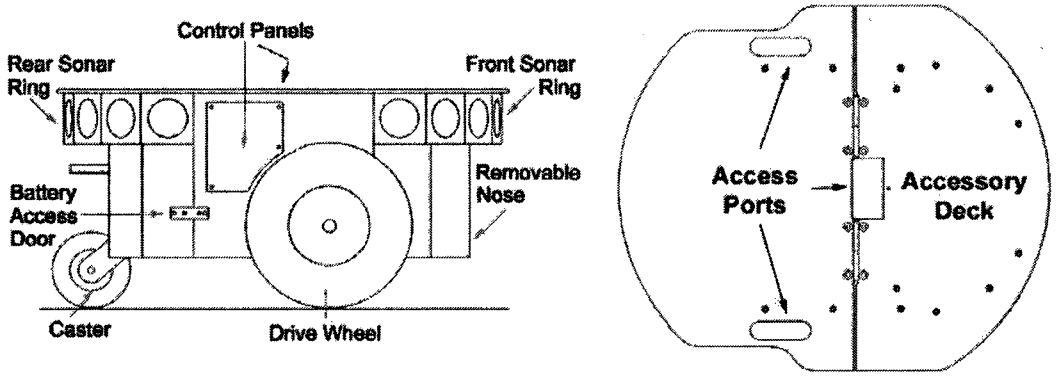


*Figure A.1- Pioneer3DX features [54]*

## PIONEER FAMILY OF ROBOT MICROCONTROLLERS

Pioneer 3DX uses revolutionary high-performance microcontrollers with advanced embedded robot control software based on the new-generation 32-bit Renesas SH2-7144 RISC microprocessor

***PORTS AND POWER***

Pioneer 3DX robot has a variety of expansion power and I/O ports for attachment and close integration of a client PC, sensors, and a variety of accessories—all accessible through a common application interface to the robot's server software, ARCOS (figure A.1). Features include:

- 44.2368 MHz Renesas SH2 32-bit RISC microprocessor with 32K RAM and 128K FLASH
- 4 RS-232 serial ports (5 connectors) configurable from 9.6 to 115.2 kilobaud
- 4 Sonar arrays of up to 8 sonar each
- 2 8-bit bumpers/digital input connectors
- Gripper/User I/O port with 8-bits digital I/O, analog input, and 5/12 VDC power
- Heading correction gyro port
- Tilt/roll sensor port
- 2-axis, 2-button joystick port
- User Control Panel
- Microcontroller HOST serial connector
- Main power and bi-color LED battery level indicators
- 2 AUX power switches (5 and 12 VDC) with related LED indicators
- RESET and MOTORS pushbutton controls
- Programmable piezo buzzer
- Motor/Power Board (drive system) interface with PWM and motor-direction control lines and 8-bits of digital input
- $I^2C$ interface with 4-line X 20-character LCD support

With the onboard PC option, the robot becomes an autonomous agent. With Ethernet-ready onboard autonomy, 3DX robot even becomes an agent for multi-intelligence work.

## *MODES OF OPERATION*

Pioneer 3DX robot can operate in one of four modes:

- Server
- Joydrive
- Maintenance
- Standalone

### Server Mode

The new Renesas SH2-based microcontroller comes with 128K of re-programmable FLASH and 32K dynamic RAM memory. In conjunction with client software running on an onboard or other user-supplied computer, 3DX lets you take advantage of modern client-server and robot-control technologies to perform advanced mobile-robotics tasks. Most users run their robot in server mode because it gives them quick, easy access to its robotics functionality while working with high-level software on a familiar host computer.

### Maintenance and Standalone Modes

This mode of operation is suitable for experiments in microcontroller-level operation of robot's functions. One may reprogram the onboard FLASH for direct and standalone operation of the robot.

### Joydrive Mode

This mode of operation is aimed to let the user drive the robot from a tethered joystick when not otherwise connected with a controlling client.

## *PHYSICAL CHARACTERISTICS AND COMPONENTS*

Weighing only 9 kg (20 pounds with one battery), the basic Pioneer 3-DX mobile robots are lightweight, but their strong aluminum body and solid construction make them virtually indestructible. The Pioneer 3-DX can carry up to 23 Kg (50 lbs.) additional weight.

Pioneer robots are composed of several main parts:

- Deck
- Motor Stop Button
- User Control Panel
- Body, Nose, and Accessory Panels
- Sonar Array(s)
- Motors, Wheels, and Encoders
- Batteries and Power

**BODY, NOSE AND ACCESSORY PANELS**

Pioneer 3DX's sturdy, but lightweight aluminum body houses the batteries, drive motors, electronics and other common components, including the front and rear sonar arrays. The body also has sufficient room, with power and signal connectors, to support a variety of robotics accessories inside, including an A/V wireless surveillance system, radio Ethernet, onboard computer, laser range finder and more.

**ACCESS PANELS**

All DX's come with a removable right-side panel through which you may install accessory connectors and controls. A special side panel comes with the onboard PC option, for example, which provides connectors for a monitor, keyboard, mouse and 10Base-T Ethernet, as well as the means to reset and switch power for the onboard computer. All models come with an access port near the center of the deck through which to run cables to the internal components.

## SONAR

Natively, ARCOS-based robots support up to four sonar arrays, each with up to eight transducers that provide object detection and range information for collision avoidance, features recognition, localization, and navigation. The sonar positions in all Pioneer 3 sonar arrays are fixed: one on each side, and six facing outward at 20-degree intervals. Together, fore and aft sonar arrays provide 360 degrees of nearly seamless sensing for the platform.

## MOTORS, WHEELS, AND POSITION ENCODERS

Pioneer 3 drive systems use high-speed, high-torque, reversible-DC motors, each equipped with a high-resolution optical quadrature shaft encoder for precise position and speed sensing and advanced dead-reckoning. Motor gear head ratios, encoder ticks-per-revolution and tire sizes vary by robot model. All Pioneer 3-DX robots come with foam-filled solid tires with knobby treads.

## BATTERIES AND POWER

Pioneer 3 robots contain up to three, hot-swappable, seven ampere-hour, 12 volts direct-current (VDC), sealed lead/acid batteries (total of 252 watt-hours), accessible through a hinged and latched rear door.

### Recharging

Typical battery recharge time using the recommended accessory (800 mA) charger varies according to the discharge state; it is roughly equal to three hours per volt per battery. The Power Cube accessory allows simultaneous recharge of three swappable batteries outside the robot.

With the high-speed (4A maximum current) charger, recharge time is greatly reduced. It also supplies sufficient current to continuously operate the robot and onboard accessories, such as the onboard PC and radios. But with the higher-

current charger, care must be taken to charge at least two batteries at once. A single battery may overcharge and thereby damage both itself and the robot.

## ACESSORIES

Pioneer 3 robots have many accessory options. For convenience, we include a description of the more commonly integrated accessories in this document. Please also refer to the detailed documents that come with the accessory.

## JOYSTICK AND JOYDRIVE MODE

Pioneer 3 robot's microcontroller has a joystick port and ARCOS contains a JoyDrive server for manual operation. Start driving your robot with a joystick any time when it is not connected with a client software program. Simply plug it into the joystick port and press the "fire" button to engage the motors.

## BUMPERS

Bump rings provide contact sensing for when other sensing has failed to detect an obstacle. The accessory rings also are segmented for contact positioning. Electronically and programmatically, the bumpers trigger digital events which are reflected in the STALL values of the standard server-information packet that ARCOS automatically sends to a connected client. ARCOS itself monitors and responds to protection triggers.

## RADIO CONTROLS AND ACCESSORIES

Pioneer 3DX platform is server in a client-server architecture. You supply the client computer to run your intelligent mobile-robot applications. The client can be either an onboard piggy-back laptop or embedded PC, or an off-board PC connected through radio modems or wireless serial Ethernet. In all cases, that

client PC must connect to the internal HOST or User Control Panel SERIAL port in order for the robot and your software to work.

For the piggyback laptop or embedded PC, the serial connection is via a common "pass-through" serial cable. Radio modems may replace that serial cable with a wireless tether. Accordingly, if you have radio modems, one is inside your robot and connected to the microcontroller's HOST serial port, and the other modem plugs into a serial port on some off board computer where you run your client software. Hence, in these configurations, there is one dedicated client computer.

Radio Ethernet is a little more complicated, but is the preferred method because it lets you use many different computers on the network to become the robot's client. If you have a PC onboard (either integrated or piggyback), it can supply the radio Ethernet connection through a PCMCIA-based wireless Ethernet card. Also a wireless Ethernet-to-serial accessory is provided which connects directly to your robot's microcontroller. It works by automatically translating network-based Ethernet packet communications into streaming serial for the robot microcontroller and back again.

Running 3DX robot through wireless Ethernet to an onboard computer is different than with the Ethernet-to-serial device. In the first case, you run your robot client software on the onboard PC and use wireless Ethernet to monitor and control that PC's operation. In the latter case, you run the client software on a remote LAN-based PC.

Accordingly, a major disadvantage of the wireless Ethernet-to-serial device is that it requires a consistent wireless connection with the robot. Disruption of the radio signal—a common occurrence in even the most modern installations—leads to poor robot performance and very short ranges of operation. This is why it is recommend to use onboard client PCs for wider, much more robust areas of autonomous operation, particularly when equipped with their own wireless Ethernet. In this configuration, you run the client software and its interactions with the robot microcontroller locally and simply rely on the wireless connection to

export and operate the client controls. Moreover, the onboard PC is often needed for local processing, such as to support a laser range finder or to capture and process live video for vision work.

## INTEGRATED PC

Mounted just behind the nose of the robot, the Pioneer 3 integrated PC is a common EBX form-factor board that comes with up to four serial ports, 10/100Base-T Ethernet, monitor, keyboard and mouse ports, two USB ports and support for floppy, as well as IDE hard-disk drives. For additional functionality, such as for sound, video frame grabbing, fire wire or PCMCIA bus and wireless Ethernet, the onboard PC accepts PC104 and PC104-plus (PCI bus-enabled) interface cards that stack on the motherboard.

### Computer Control Panel

User-accessible communication and control port connectors, switches and indicators for the onboard PC are on the Computer Control Panel, found on the right side panel of the DX or in the hinged control well next to the user controls of the AT. The controls and ports use common connectors: standard monitor DSUB and PS/2 connectors on the mouse and keyboard. The Ethernet is a 10/100Base-T standard RJ-45 socket. The ON/OFF slide switch directly controls power to the onboard PC—through Main Power, unlike some earlier versions of the onboard system which included a delayed power shutdown. The PWR LED lights when the computer has power. The HDD LED lights when the onboard hard-disk drive is active. The RESET button restarts the PC.

### PC Networking

The RJ-45 connector on the Computer Control Panel provides wired 10/100Base-T Ethernet networking directly with the onboard PC. With the purchased option, we also install a PCMCIA adaptor card on the PC's accessory stack and insert a wireless Ethernet card in one of its slots. The wireless Ethernet antenna sits atop the robot's deck. To complete the wireless installation, you will

need to provide an Access Point to your LAN (comes as an accessory with most units). Attach the Access Point to one of your LAN hubs or switches. No special configuration is required.

# VITA AUCTORIS

Ahmad Soleimani was born in 1972 in Baghdad, Iraq. He obtained a Bachelor degree in Computer software Engineering from the department of Electrical and Computer Engineering of Shahid Beheshti University- Tehran, Iran in 1995. He is currently a candidate for the Master's degree in Computer Science at the University of Windsor and hopes to graduate in Spring 2010.