How Albot$_1$ Computes Its Perceptual Map

Md. Zulfikar Hossain

A thesis submitted to
Auckland University of Technology
in fulfilment of the requirements for the degree of
Doctor of Philosophy (PhD)

Primary Supervisor:
Professor Wai-Kiang (Albert) Yeap

July, 2014

School of Computer and Mathematical Sciences

To my parents

Table of Contents

List of Tables

List of Figures

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgments), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Md Zulfikar Hossain
AUT, New Zealand.

Abstract

This thesis describes the implementation of Albot_1, the second in the series of Albots. Albots are robots created in Centre for Artificial Intelligence Research (CAIR) Laboratory at AUT for investigating hard problems in cognitive science concerning spatial cognition. To create an Albot, a computational theory is first proposed and then implemented on a mobile robot. Its behavior is then studied to gain further insights into the underlying process. With Albot_1, the problem being studied is: how an imprecise and incomplete map is computed without integrating every successive view of the environment. Albot_1 is based on Yeap's (2011a) computational theory of perceptual mapping.

That this problem is both worth studying and hard is evident in the controversial debate among psychologists as to the nature of such a map. Despite much interest, the notion of a cognitive map remains vague and controversial. Furthermore, robotics research has shown that the traditional process of integrating every successive view to form a map is highly susceptible to cumulative errors coming from the sensors. These errors must be corrected in order to produce a useful map, but in doing so, what is produced is a precise and complete map. Humans don't have such a map in their head but then, how and what kind of map is produced?

Yeap's (2011a) theory suggests that a map is formed from integrating views representing local environments visited. In his model, viewers don't track their position in the map, nor are errors from the sensors corrected. Rather, they attend to landmarks and remember the local environment that they are about to explore. Describing these local environments using a single global frame of reference gives one a map of

the entire environment traversed.

The theory was tested with a robot equipped with a laser and an odometer. Six experiments were conducted to evaluate both the process and the map computed. Specifically, the experiments evaluated: (i) Albot_1's ability to compute a map in an office-like environment, (ii) Albot_1's ability to orient itself, (iii) Albot_1's ability to handle errors, (iv) Albot_1's ability to compute its map using continuous motion, (v), and (vi) Albot_1's performance compared with SLAM-based algorithms.

The results provide confirmatory evidence that the process is both robust and useful. It successfully captures the overall shape of the environment traversed as long as no parts are being re-visited. Once a familiar part is being re-visited, the previous experience of that part of the environment is wiped from the map and overridden. The map computed is thus both a transient and a dynamic map. It represents a trace of one's experience through the environment rather than a complete map of the environment experienced. Having successfully computed a transient map, I also investigated how a more enduring map, in the form of a topological network of places, could be built. A further three experiments were conducted to evaluate Albot_1's "cognitive map".

The successful implementation of Albot_1 introduces a new paradigm for robot mapping and provides significant insights into how Yeap's (2011a) perceptual mapping works.

Acknowledgments

I am grateful to the Almighty, most Merciful, and most Gracious, who gave me the strength to survive.

I offer my sincerest gratitude to my supervisor, Professor Wai Yeap, who gave me the opportunity to work in his Lab. This thesis would not be like this without his guidance, criticism, support, encouragement, and motivation for the last four and a half years; the most valuable advice being: "think..think..think..", which I will remember in the rest of my life. Thank you Professor.

I would like to thank the members of the thesis committee for their critical reviews in improving my thesis.

I would like to thank my lab mate, Lonnie Wilkerson and his wife, Hollie for their assistance in improving my writing.

I would like to thank my friends and colleagues at AUT for their support and encouragement for the last four years. In particular, I would like to express my gratitude to Dr. Zati, Dr. Waseem, Dr. Issam, Dr. Thamilini, Dr. Richard Liu, Md. Akbar Hossain, M H Islam, Khaled Mahbub Morshed, Thomas, for sharing their experience with me to help pass the difficult times.

I would like to express my gratitude to my family for their continuous support to overcome all the struggles for my entire life.

This research was mostly supported by the BuildIT PhD scholarship, University of Waikato and partially supported by School of Computer and Mathematical Science, AUT. I recognize that this research would not have been possible without their financial assistance and express my gratitude to those organizations.

Auckland, New Zealand
July 2014

Md Zulfikar Hossain

Chapter 1

Introduction

This thesis describes the implementation of Albot$_1$, a robot designed to investigate how cognitive species might compute a "map" of their environment. Albot$_1$ is part of an on-going series of robots being designed in our laboratory for studying spatial cognition. The first in the series is Albot$_0$. This robot is designed to investigate the imprecise nature of cognitive maps and in particular, how such a map could be useful (Yeap, 2011b; Wong, 2008). Albot$_1$ is the second in the series. It is designed to investigate how an inexact and incomplete map is computed from integrating successive views of the environment.

Albots are robots created individually to study problems that have confounded behavioral researchers interested in spatial cognition. These problems arise because of the conflicting interpretations of these researchers' findings. Consequently, the exact nature of the underlying process is in doubt. To create an Albot, a computational theory is first proposed and then implemented on a robot. Its behavior is then studied to shed light onto the process itself as if the Albot is one of the natural species performing the task. One novelty in creating an Albot is that the cognitive problem identified is solved at its level by using its own embodiment.

Spatial cognition is concerned with the derivation and utilization of spatial knowledge. Initially, and at the perceptual level, one is concerned with computing a "map" of one's environment directly from one's sensors. This sub-process is often referred to as spatial (or perceptual) mapping. Researchers interested in spatial cognition often refer to one's spatial knowledge as a "cognitive map", a term coined by Tolman (1948). Tolman proposed the idea as an alternative explanation to the then popular stimulus response strategy. This accounted for what he considered as the insightful behavior of rats in mazes, notably their ability to find short cuts. Since then, numerous psychological experiments, with rats (Olton, 1977; Roberts, 1979; Zoladek & Roberts, 1978), humans (Carr & Schissler, 1969; Siegel & White, 1975), and other animals (Cartwright & Collet, 1982; Gould, 1987; Maury, Mauch, Hammer, & Bingman, 2010; Presotto & Izar, 2010; Rosati & Hare, 2012) were designed to investigate the nature of cognitive maps. Researchers from other disciplines found the idea interesting too. For example, Lynch (1960) conducted a study of residents' memories of their cities and found that they share common elements such as landmarks, paths, edges, and regions. The "images" of their city are their cognitive maps. Since then, geographers and urban designers have conducted numerous studies on human cognitive maps (Downs & Stea, 1973; Moore & Golledge, 1976). O'Keefe and Nadel (1978) identified the hippocampus as the neural site for computing a cognitive map. Their work sparked a debate of, and a search for, the neural substrate for cognitive maps (Kumaran & Maguire, 2005).

Despite the huge interests, the notion of a cognitive map remains vague and controversial. In Bennet's (1996) review of the notion of a cognitive map, he concluded that: "The term 'cognitive map' has been used to describe a variety of concepts. Many

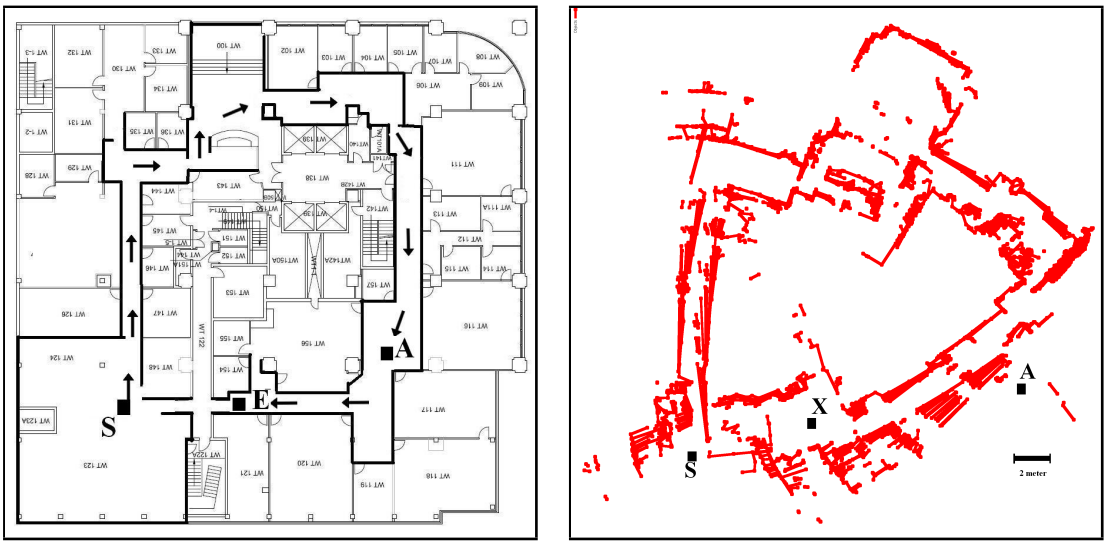
of these are contradictory. Others are too imprecise to make clear behavioral predictions (p. 223)". He noted that two main definitions are often used in the literature. One, according to Tolman (1948) and O'Keefe and Nadel (1978), is a representation that enables novel short-cutting to occur. The other, according to Gallistel (1990), is that a cognitive map is any representation of space. Both, however, fail to highlight the map-like property of the term "cognitive map". Bennet suggests that "the cognitive map is no longer a useful hypothesis for elucidating the spatial behaviour of animals, and that use of the term should be avoided (p. 223)". While the term is often used to refer to one's representation of one's environment, what is unclear is its map-like property that supposedly distinguishes it from other known knowledge of one's environment.

Yet as we move, we remember where we are in our immediate environment. Our abilities to point to places passed and distant objects and to draw sketch maps of our local environment suggest that a map is being computed. Researchers studying such human spatial ability often assume that successive views are integrated to form a map. Since our initial view is described using an egocentric co-ordinate system, it is further assumed that the integration of such views will initially produce a map that uses an egocentric co-ordinate system. Such a map is referred to as an egocentric map and, as we move, the egocentric locations of every object in the environment are updated. The latter step imposes a limit on the map created. It is often suggested that a much larger and enduring map of the environment is somehow created from the egocentric map (Sholl, 2001; Burgess, 2006). This then becomes our cognitive map and uses a non-egocentric frame of reference, commonly referred to as an allocentric frame of reference. Recently, Wang and Spelke (2000, 2002) challenged this idea,

claiming that no allocentric map is ever computed as part of our perceptual process. They claim that the allocentric map in our head is a result of our symbolic reasoning about the environment.

Robotics researchers have been studying the same process: integrating successive views to form a map of the environment. However, they quickly realized that the errors in the sensors could easily distort the map computed (for example see figure 1.1). To correct these errors, they need to simultaneously localize and map as the robot moves. Doing so allows them to track and correct the errors to produce a precise map. Robotics researchers refer to their mapping problem as SLAM, a simultaneous localization and mapping problem. Over the past two decades, they have made significant advances in developing probabilistic solutions for the problem (Durrant-Whyte & Bailey, 2006).

While the robotics solution is an elegant one, it is not suitable for explaining how cognitive agents compute their map. In fact, and as Yeap (2011a) argued, their solution casts doubt that cognitive agents compute their map via integrating every successive view. This is because if one were to do so, one must correct the sensor errors prior to producing a useful map. The resulting map is necessary precise and robotics researchers are able to use the map to recognize places re-visited via loop closing. However, humans don't compute a precise map and close loops. Furthermore, errors in human vision are worse than those found in robot sensors (Hurlbert, 1994; Fermuller, Cheong, & Aloimonos, 1997; Glennerster, Hansard, & Fitzgibbon, 2009) and therefore if the same approach is used, it would become more essential that errors from sensors are corrected. Instead they rely on complex recognition routines to recognize places re-visited.



(a) (b)

Figure 1.1: (a) A test environment and (b) a map computed by using simple mathematical transformation. During this experiment, the robot traveled approximately 100 meters from point S to point E following arrow directed route as shown in figure a. In figure b, location A appears at location X (perceptual error approximately 10 meters). Reproduced from Yeap (2011a).

The problem we attempt to answer in this thesis is: How do humans, in particular, and animals, in general, compute an initial map of their environment so they can orient themselves in it? Using Albots, this problem is studied in its abstract form. The precise question we ask in creating Albot_1 is:

> How could an autonomous agent, given some sensors that allow it to compute distances of objects in view, calculate how far it has turned and/or moved, and recognize similar objects in successive views, compute a useful map that is imprecise and incomplete?

By emphasizing that the map computed is imprecise and incomplete, Albot_1 is essentially solving a problem similar to that solved by humans. Observing how Albot_1

computes its map, it is realized that much could be learned as to how cognitive agents compute their own map. This thesis focus only on the implementation of Albot_1.

As noted above, creating an Albot requires a computational theory. Albot_1 is created based on Yeap's (2011a) computational theory of perceptual mapping. Yeap argues that a view makes explicit not just what and where things are but also an environment that affords venturing into. By assuming that the physical environment one lives in is stable, the theory argues that such a view forms the cornerstone representation for building one's spatial knowledge about the physical environment. When one remembers a view, one obtains a "map" of the local environment before one ventures into it. As one moves from one such local environment to another, these local maps are pieced together to form a global metric map of the larger environment traversed. As such, the map is not updated continuously with each subsequent view; one updates only as one leaves a local environment and enters another local environment. There is also no explicit tracking of one's position within the map. Instead, one tracks familiar objects in successive views. These objects were those seen earlier, in the view remembered, and their positions in the map are thus known. With them in sight, one could triangulate one's approximate position in the map when needed. These objects, thus function as landmarks in the current local environment and if none of them are in view, one is deemed "lost". As such, the map is updated prior to all the landmarks disappearing from view.

During updating, and as mentioned above, the viewer's position in the map is triangulated using the remaining landmarks in view. The current view is then transformed and added to the map. There is no explicit boundary to distinguish between different local environments in the map. The resulting map is thus a global metric

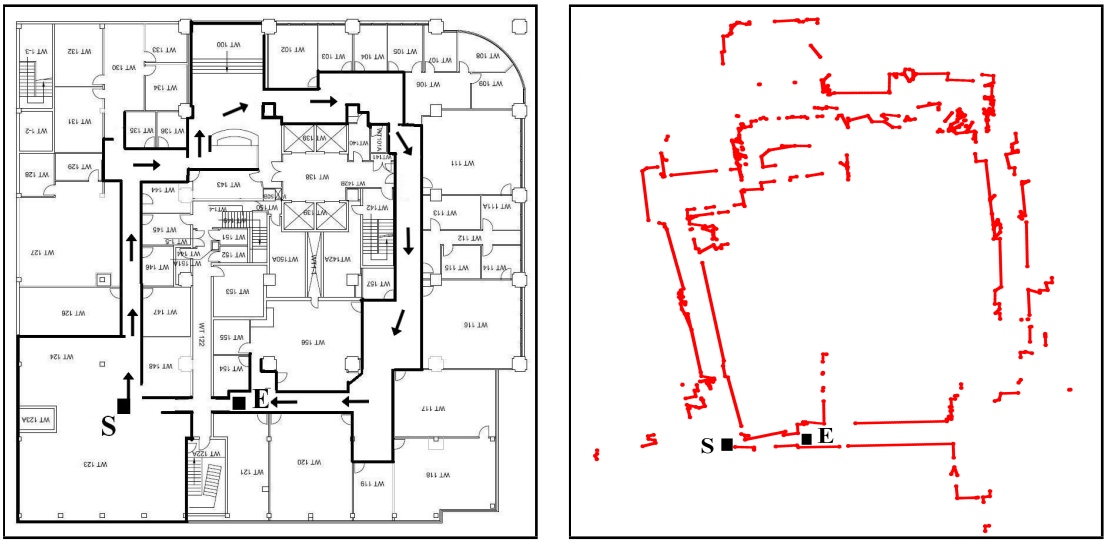
map. Information from the incoming view is not matched with information in the map during updating. The part of the map occupied by the incoming information is simply deleted and replaced by information from the incoming view. Consequently, the map produced is incomplete and never closed. Recognition of the places re-visited is, in general, not achieved at the perceptual level; it is done at the conceptual level.

At the perceptual level, the perceptual map is thus computed for two reasons. First, it makes explicit the local environment that one is about to explore. Second, it makes explicit the larger environment that one has traversed. The former is achieved by adding the current view to the map as is i.e. without matching what is in it with what is in the map. The latter is achieved by constructing an allocentric map consisting of local environments visited. Note that there is no error correction of the distortions due to sensors and the map produced is thus imprecise. The global map is distorted in the sense the viewer's position is not where it should be. Figure 1.2b shows a perceptual map computed using the theory and as implemented in this thesis.

In this thesis, I implemented the theory described using a robot equipped with a laser and an odometer. Albot$_1$ has, as input, the key information available to other species for computing a cognitive map, namely distance of surfaces perceived and proprioceptive information about its own movements. Albot$_1$ needs to track landmarks in successive views and builds a global map consisting of local environments. The latter is achieved by adding a new view to the map whenever very few landmarks are in view. However, without vision, Albot$_1$ lacks a description of what is perceived and thus could not recognize landmarks in view by comparing their properties. Consequently, a simple "recognition" algorithm is implemented for Albot$_1$: it uses co-ordinate transformation of successive views to identify which landmark surfaces in

the current view match the surfaces in the previous view due to their close proximity and orientation in the expected locations. If they match, then these landmarks are in view. If not, it is assumed that the landmark has disappeared from view. This simple strategy is found to be adequate for Albot$_1$ to track landmarks in view.

While it has been emphasized that the creation of an Albot would lead to a better understanding of the underlying cognitive process, the scope of this study is limited to the algorithmic aspects of Albot$_1$ rather than its contribution to cognitive mapping. Consequently, five subsequent experiments were designed to evaluate how well the algorithm performs and its limitations. In the first experiment, the goal is to show what kind of a map is computed by Albot$_1$. Will the map be distorted to the extent that it is not useful? Figure 1.2b shows that the map computed maintains an overall shape of the environment traversed (as compared with figure 1.1b). Subsequently, two



(a) (b)

Figure 1.2: (a) The test environment and (b) a perceptual map computed from the data derived from exploring the environment as shown in figure a. (For comparison, this data is the same as that used to generate the output as shown in figure 1.1b)

more experiments were conducted to show how good is the map computed - one measures Albot_1's ability to orient itself in the environment using its map and the other measures the effect of errors on Albot_1's mapping process. All three experiments were conducted with Albot_1 exploring its environment in a controlled start-stop manner. In the next two experiments, Albot_1 explores its environment in a controlled continuous fashion and logs its input data. The data is then used to test Albot_1's ability to re-compute its map (experiment 4) and to compare Albot_1's performance with some well-known SLAM-based algorithms (experiments 5) and to compare Albot_1's performance using dataset used in robotics research (experiment 6).

The rest of the thesis is organized as follows. Chapter 2 provides a review of both the psychological and robotic research on the mapping problem that Albot_1 is designed to solve. In addition, other cognitively inspired approaches and the Albots approach are also reviewed. Chapter 3 provides detailed implementation of Albot_1's perceptual mapping algorithm together with the six experiments described above. Four algorithms utilized by Albot_1 are described. Chapter 4 describes an extension of Albot_1's ability to compute a "cognitive" map of its environment and two more new algorithms are added. Chapter 5 concludes the thesis with a general discussion of the results obtained and future works.

Chapter 2

Background and Literature Review

Creating an Albot requires one to first identify a key question about the underlying process in cognition and then use the robot's own physical embodiment to implement a solution to the problem. The latter, if successful, turns the robot into an Albot whose behavior could then be studied along with other species to gain a deeper understanding of the process itself.

One question concerning spatial cognition that we have identified is:

> How could an autonomous agent, given its own sensors, compute a useful map that is *imprecise and incomplete?*

Many cognitive researchers, over a period of 60 years, have argued that humans and other species compute an imprecise and an incomplete "map" of their environment. Yet, there is still no clear idea what kind of a map is computed. It is believed to be something we compute from integrating successive views of the environment as it is being explored. Yet, robotics researchers, in their attempt to implement a similar process for mobile robots over a period of 20 years, fail to produce such a map. Instead, they quickly realize that sensor errors will distort the map computed and concluded that in order to have a useful map, these errors must be corrected and

subsequently, a precise map is produced. This situation creates what Yeap (2011a) describes as a perceptual paradox: to create a useful map, we need to correct errors due to our senses, but if we do, we get a map that we don't have.

In this section, some relevant works from these two bodies of research are briefly reviewed to provide background to the identified paradox. In particular, section 2.1 reviews psychological studies that investigate how human's spatial memory is updated as he/she moves in the environment and section 2.2 reviews one of the most significant approaches in robot mapping, namely SLAM (simultaneous localization and mapping). Section 2.3 reviews other cognitively inspired robot mapping approaches as well as the progress in developing Albots to date and section 2.4 concludes this chapter.

2.1 Psychological Models

The idea that cognitive species compute a mental or cognitive "map" has been popular among cognitive researchers (Tolman, 1948; O'Keefe & Nadel, 1978), including those working on insects (Wehner & Menzel, 1990), but notwithstanding some later criticisms of the idea in general (Bennett, 1996). Many of the earlier studies in cognitive mapping focused on what is remembered after one has become familiar with the environment. For animals, the test is often about their ability to use the map to find novel/shorter routes to a goal (Tolman, 1948; Gould, 1987; Dyer, 1991) and their ability to recall where things are (Jacobs & Limant, 1991). For humans, the focus is on how a map that is laden with our rich interpretations of our environment is computed. For example, researchers are interested in how landmarks are utilized (Siegel & White, 1975; Allen, Kirasic, Siegel, & Herman, 1979; Spetch, Cheng, MacDonald,

& Linkenhoker, 1997; Cornell, Heth, & Alberts, 1994) and how hierarchies of places are formed (Hirtle & Jonides, 1985). Many computational models are produced that show how a rich cognitive map is computed (Kuipers, 1978; Kuipers & Byun, 1988). Such early work paid little attention to the perceptual end of the process (Yeap, 1984).

This has changed in recent years and more studies have been conducted to investigate how one remembers the spatial layout of environments from direct experience and how one's memory of the current local environment could affect the performance of spatial tasks. For example, Ishikawa and Montello (2006) studied how their subjects' spatial knowledge changes after each session of being driven along two routes in an unfamiliar environment. They found that "some participants' knowledge improved fairly continuously over the sessions. However, most participants either manifested accurate metric knowledge from the first session or never manifested accurate metric knowledge (p. 93)". In another experiment, Cheng (1986) tested rats in place finding tasks in a rectangular environment with distinct featural panels in the corner and found that the rats relied more on the shape of the room for orientation than the overall arrangement of non-geometric features. Since Cheng's seminal work, many researchers have found that other animals rely on the geometry of the local environment for orientation (Sovranoa, Bisazzaa, & Vallortigarab, 2002; Lee & Vallortigara, 2012; Sovrano, Rigosi, & Vallortigara, 2012), for review, see (Cheng & Newcombe, 2005).

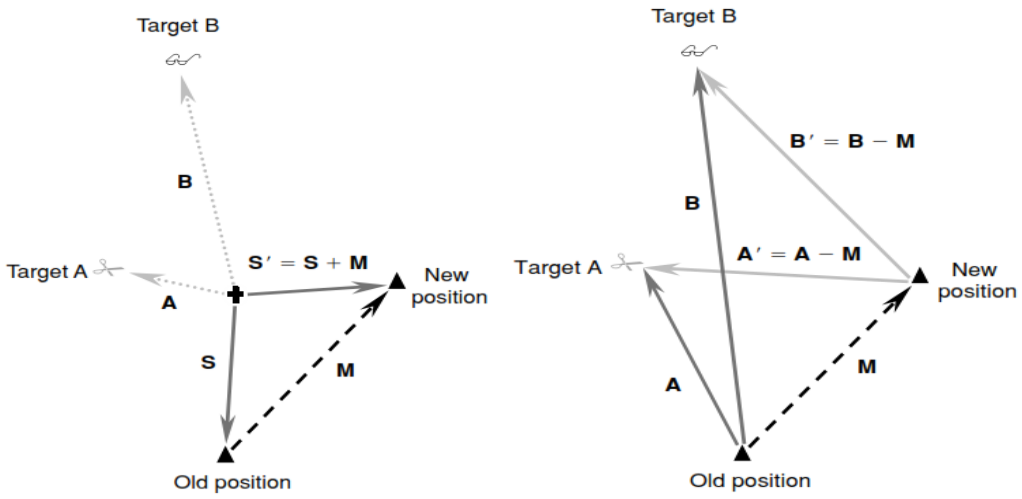
For this review, we describe three experiments that were designed to investigate how spatial memory is updated as one moves. Interestingly, these researchers assume that the mapping process computes a map of the environment by continuously

transforming the coordinates of one view to the next. They question whether the map computed uses an egocentric frame of reference or a non-egocentric frame of reference, the latter often referred to as an allocentric frame of reference. Intuitively, and given that one begins with an egocentric description of the environment, the initial map computed is an egocentric map. It is argued that such a map is daunting to maintain as the map grows in size (Burgess, 2006) and consequently, it is believed that information in it will later be transferred to a more enduring non-egocentric map. Until recently, this dual model of the mapping process has been widely accepted with notable proponents. Wang and Spelke (2002) argue that no allocentric map is ever computed, whereas Mou and McNamara (Mou & McNamara, 2002; McNamara, 2003) argue that no egocentric map is computed. These works are reviewed below. It should be noted that, as we take a symbolic approach in our study, non-symbolic approaches to cognitive mapping will not be reviewed. For the latter, see a recent review in Hartley, Lever, Burgess, and O'Keefe (2013).

2.1.1 Wang and Spelke's (2000) Model

Using a disorientation paradigm, Wang and Spelke (2000) tested whether human navigation depends on enduring allocentric representations, or on momentary egocentric representations. They suggested that the configuration error after disorientation will reveal which representation is being used. Configuration error provides measurement of the relative accuracy of pointing to different objects and is defined as the standard deviation of all individual object's errors. Their hypothesis is as follows: Disorientation will have different effects on enduring allocentric and dynamic egocentric spatial

representation of objects. If humans remember the locations of objects allocentrically, pointing-accuracy to objects after disorientation will be the same. In contrast, if humans remember the location of objects egocentrically, then disorientation configuration error will be increased. This is because, in allocentric representation, only one update (view's position) is necessary with respect to the reference system, whereas in egocentric representation, update is necessary for all individual objects with respect to view's new position (see figure 2.1).



(a) (b)

Figure 2.1: An illustration of spatial updating for (a) an allocentric model in which the central cross represents an external reference system. The target locations are represented as A and B and remain the same as the observer, represented by the filled triangle, moves from the old position to the new position. Spatial updating involves computation of the observer's new position, S' = S + M. (b) an egocentric model in which the target positions (A and B) are represented relative to the observer. Spatial updating involves computing the new egocentric positions (A' and B') of the objects as the observer moves (A' = A-M; B' = B-M). Reproduced from Wang et al. (2006).

They tested their hypothesis using seven experiments under different conditions. For example, in one experiment, subjects learned locations of six objects situated

around a chamber as shown in figure 2.2a. In the testing phase, subjects pointed to six objects from the center of the chamber with eyes-open (served as based line pointing), eyes-closed after brief rotation (served as orientation pointing), and eyes-closed after extensive rotation (served as disorientated pointing). They found that configuration error increases significantly after disorientation. In another experiment, subjects learned the location of objects and the corners of a rectangular room as shown in figure 2.2b. They found that, disorientation has no effect on pointing to the corners. From these findings, Wang and Spelke argued that humans compute a dynamic egocentric representation of objects locations.

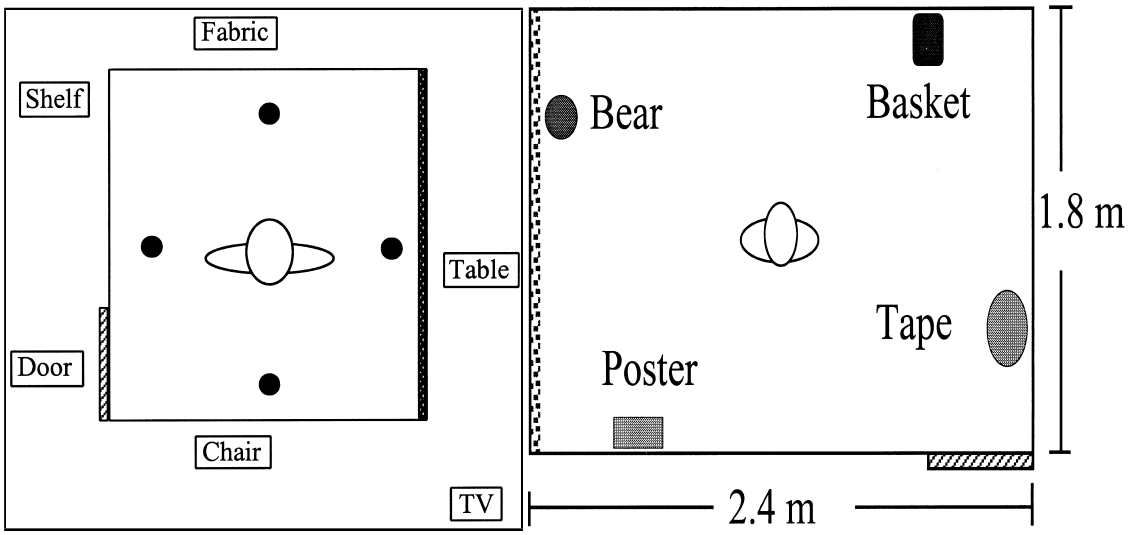


Figure 2.2: (a) The layout of objects used for the first experiment. (b) An overhead view of the rectangular room used for pointing to corners. Reproduced from Wang and Spelke (2000).

Based upon the above findings and a brief review of research on navigating animals, from ants to primates, Wang and Spelke (2002) conclude that even humans do not compute an allocentric map directly from perception. However, they do have

allocentric maps in their head but these are the results of their symbolic reasoning about their environment rather than being computed directly from their perception of the environment.

2.1.2 Waller and Hodgson's (2006) Model

One of the difficulties with Wang and Spelke's model is that it is impossible to maintain a purely egocentric map for one's total experience of the environment (Burgess, 2006), since an egocentric map involves the updating of all object's positions for every step. Consequently, many have argued against it (Mou & McNamara, 2002; McNamara, 2003; Waller & Hodgson, 2006; Frankenstein, Meilinger, Mohler, & Bulthoff, 2009). One interesting work is that of Waller and Hodgson (2006) who used Wang and Spelke's disorientation paradigm to show that it supports both an egocentric and an allocentric representation. They developed a "switch interpretation" to explain Wang and Spelke's (2000) findings, namely as participants do not forget relative directions of objects completely, they learn both representations. Disorientation, however, forces them to depend on the less precise, enduring representation instead of the more accurate, transient egocentric representation.

To test their idea, they designed an experiment similar to Wang and Spelke's (2000) experiment, where subjects learn location of six objects situated around a square chamber as shown in figure 2.3a. Like Wang and Spelke's (2000) experiment, in testing phase, subjects were asked to sit at the center of the chamber and perform egocentric pointing (point to different objects from their current position) which involves accessing spatial information from a transient egocentric representation. In

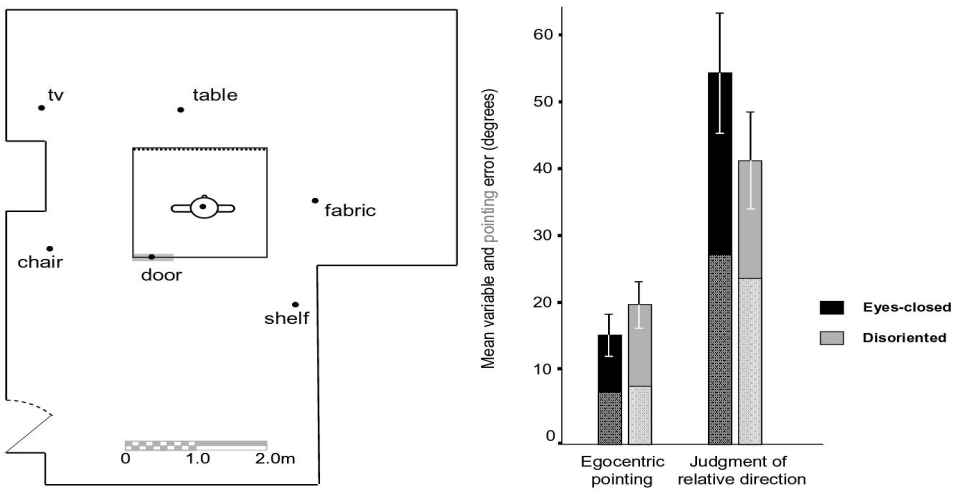
addition, like Mou and McNamara's (2002) experiment, subjects were asked to perform judgments of relative directions (JRDs) (Imagine that you are at the TV, facing the door. Point to the fabric.) which involves accessing spatial information from an enduring spatial representation.

To measure subjects' knowledge of objects locations, they calculated pointing errors for both tasks before and after disorientation as shown in figure 2.3b. As the egocentric pointing error was much smaller than that obtained by Wang and Spelke's (2000) experiment, and JRDs pointing error decreased after disorientation, they suggested that human spatial representation involves two systems a transient egocentric representation and an enduring spatial representation, switching happens after disorientation. In addition, as JRDs pointing error increased significantly more than the egocentric pointing errors, they argued that the transient egocentric representation is more precise than the enduring spatial representation.

2.1.3 Mou and McNamara's (2002) Model

Taking inspiration from human shape perception (Rock, 1973), Mou and McNamara (Mou & McNamara, 2002; McNamara, 2003) proposed yet another interesting spatial model, one that is purely allocentric. Their hypothesis is that if humans remember the location of objects egocentrically, then pointing accuracy would be better on the imagined heading parallel to the study view than all other imagined heading. In contrast, if humans remember the location of objects allocentrically, then pointing accuracy would be better on the imagined heading parallel to intrinsic axis than all other imagined heading.

They designed three experiments to test their hypothesis. In one experiment,

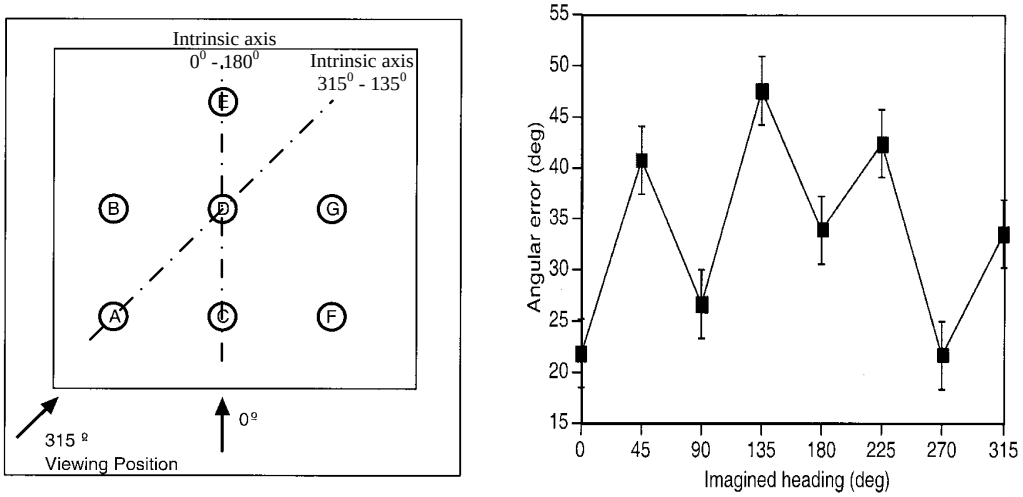


(a) (b)

Figure 2.3: (a) The layout of objects used for experiment 1. (b) Mean variable error for egocentric pointing and judgments of relative direction (JRDs) in the eyes-closed and disoriented phases. The portion of variable error that can be accounted for by pointing error is shown in the bottom region of each bar. Reproduced from Waller and Hodgson (2006).

they used the layout of letters and disks as shown in figure 2.4a. To make an intrinsic axis more noticeable, they induced cues such as unique color for each column, the same color for the same column, rectangular mat, and the wall of the room. The main purpose of this experiment was to test whether subjects learn the location of objects with respect to egocentric or intrinsic (allocentric) frame of reference. In the learning phase, all subjects were asked to learn object locations with respect to an intrinsic axis 0° - 180° from 315° viewing position. In the testing phase, participants were taken to another room and asked to point to objects from different imagined headings (e.g. imagine standing at the C and facing the D (imagined heading 0°), then point to G). Figure 2.4b shows the mean angular error in pointing to the objects for different imagined headings. Subjects pointed more accurately from the imagined

heading 0° (parallel to intrinsic axis) than imagined heading 315° (parallel to view direction). In addition, their performance was better from the imagined headings of 90°, 180°, and 270° aligned with an environmental cue (such as a mat or wall of the room) than from the imagined headings of 45°, 135°, and 225°. From these findings, they concluded that subjects represented objects using an intrinsic frame of reference which were selected based on the properties and layout of objects and the environment structure itself (such as a rectangular mat or the wall of a room).



(a) (b)

Figure 2.4: (a) The layout of letters and the disk used for experiment 1. (b) Angular error in pointing to the objects for different imagined headings. Reproduced from Mou and McNamara (2002).

Note that in their model, it is not suggested that no egocentric representation is computed. During locomotion through a familiar environment, an egocentric representation is updated as long as there is perceptual support, whereas an environmental representation is updated when subsequent learning experiences are aligned with salient axes. The environmental representation keeps track of the location and orientation and considers the observer as any other object in the environment. This helps

to regain position and orientation during lost situations (Mou, McNamara, Valiquette, & Rump, 2004).

2.1.4 Conclusion

Researchers interested in how humans compute a map of their immediate environment adopted a simple process model: information from each view is integrated to form a map. They then investigate what frames of reference are used to describe the map computed and they found serious inconsistencies in their explanation. We now look at an attempt to implement such a process using robots.

2.2 Robotics Approach: Simultaneous Localization and Mapping (SLAM)

Robotics researchers have also been investigating how robots can compute a map using the same process proposed by the psychologists: integrate successive views to form a map of the environment. However, because robotics researchers must implement the process, they can uncover technical issues that psychologists had no need to consider. To begin with, in implementation, the robot needs to localize itself after each move; otherwise it is not possible to integrate the current view as part of the existing map. Consequently, they refer to this problem as simultaneous localization and mapping, or, in short, SLAM. They discovered a serious problem: errors in the sensors can cause the map to be seriously distorted. To get a useful map, these errors, the researchers argued, must be corrected. Robotics researchers have since been developing algorithms to do so. The best of these algorithms are probabilistic

in nature (Thrun, Burgard, & Fox, 2005).

A standard graphical model of the SLAM problem is shown in figure 2.5. It shows the dependencies of different variables in the SLAM problem. Shaded nodes represent data variables which a robot collects with its sensors (range sensor and odometer) at a different time and location. Observation data Z_t consists of position coordinates of objects which are situated within the robot's view at time t. Odometer data u_t represents translation and rotation of the robot from its previous position X_{t-1}. X_t represents the best possible location of the robot at time t. Map m represents the positions of all seen objects so far in a global coordinate frame. In SLAM, the problem of integrating new observation to the map is referred to as the data association problem (Bar-Shalom & Fortmann, 1988; Montemerlo & Thrun, 2003). The goal of probabilistic solutions of the SLAM problem is to find the best possible location of the robot X_t such that the misalignment of the present observation data Z_t can be minimized for integration with map m for updating.

Over the last two decades, SLAM based algorithms have been implemented to solve data association problems for mapping different environments such as indoor (Davison, Reid, Molton, & Stasse, 2007), outdoor (Nuchter, Lingemann, Hertzberg, & Surmann, 2007), underwater (Fairfield, Kantor, & Wettergreen, 2007), underground (Huber & Vandapel, 2006), and airborne (Kim & Sukkarieh, 2007); as well as different modes of the environment such as static (Kretzschmar, Grisetti, & Stachniss, 2010) and dynamic (Minguez & Montano, 2005), and also in different dimensions such as 2D (Wulf, Arras, Christensen, & Wagner, 2004) and 3D (Pathak et al., 2010).

In this section, three state-of-the-art SLAM based algorithms are described. All three algorithms are popular in the SLAM community because they are efficient at

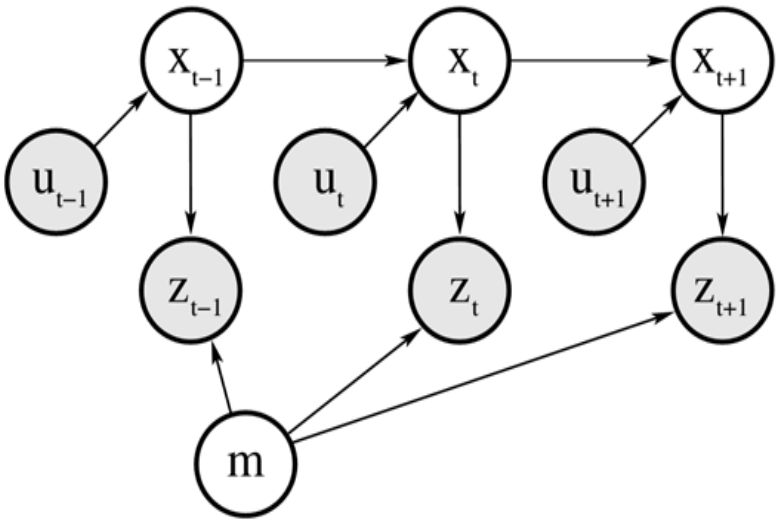


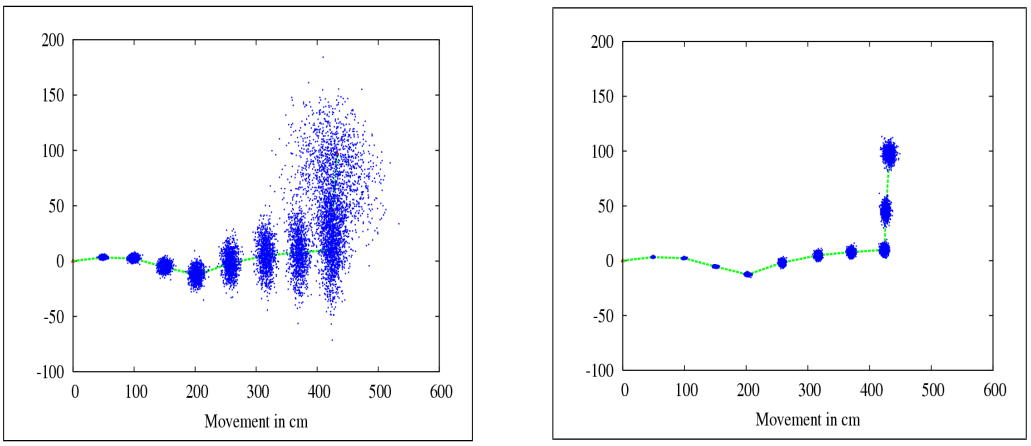
Figure 2.5: Graphical model of the SLAM problem. Arcs indicate causal relationships, and shaded nodes are directly observable to the robot. In SLAM, the robot seeks to recover the unobservable variables (X_t and m).

mapping in terms of accuracy and computational time, and their implemented codes are available online (Stachniss, Frese, & Grisetti, 2007). Two of the algorithms are based on particle filter methods and the other is based on graph optimization.

2.2.1 An Efficient FastSLAM Algorithm

Hahnel, Burgard, Fox and Thrun (2003) proposed an efficient FastSLAM algorithm based on Rao-Blackwellized particle filter. There are 4 basic steps in a particle filter based algorithm: sampling, assigning weights to the new samples, re-sampling, and map estimation. For each new robot position, new samples are introduced to estimate the position of the robot (step 1) and each estimate is then given a weight that predicts how good the estimation is (step 2). Re-sampling removes some poor samples and a

map is formed based on all robot positions and observations so far. In FastSLAM, a new step, scan matching, is added before the four basic steps. The function of this step is to improve the odometer data using all previous observation data (except the recent one) and all odometer data (including the recent one). As a result, it could generate fewer and more accurate samples. In figure 2.6a, one could see the explosion of samples as the robot moves forward. This is needed to track the robot's position. With scan matching, figure 2.6b shows fewer samples are needed.

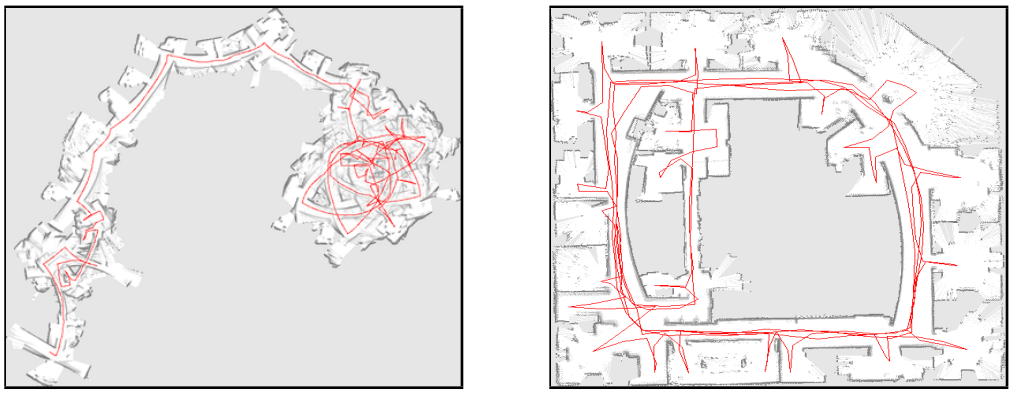


(a) (b)

Figure 2.6: Particle densities obtained with the models, (a) without using Scan Matching and (b) with Scan Matching. Reproduced from Hahel et al. (2003).

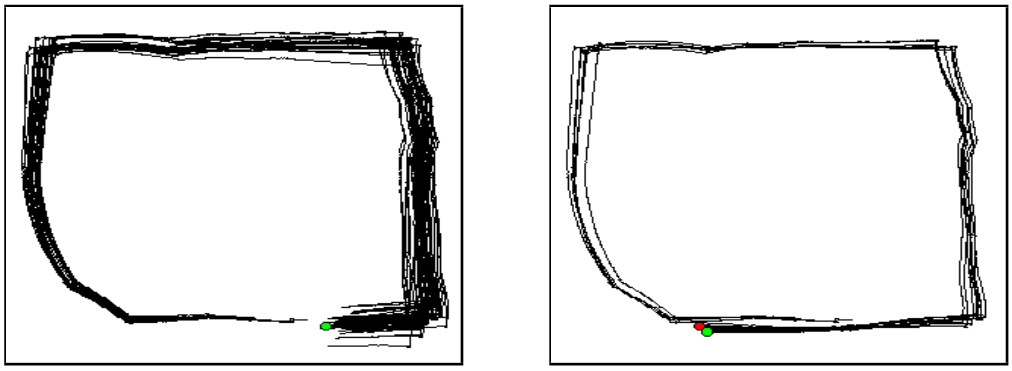
The efficient FastSLAM (Hahnel et al., 2003) algorithm has been implemented on a Pioneer 2 robot equipped with a SICK LMS laser range-finder and tested in a considerably large-scale (28m $\times$ 28m) office-like indoor environment (Intel Research Lab, Seattle, WA). Figure 2.7a shows the map computed only relying on the raw odometer data without any error correction model. Figure 2.7b shows the map computed using the efficient FastSLAM algorithm in real-time with 100 particles. Due to particle filtering, this algorithm provides multiple hypotheses during loop closing

(filter with 100 particles computes 100 robot trajectories providing 100 possibilities to close loop as shown in figure 2.8). Particle filter together with scan matching step enables this algorithm to compute an accurate map for large-scale environments with multiple loops (for detail results see (Hahnel et al., 2003)).



(a) (b)

Figure 2.7: Map of the Intel Research Lab dataset (a) computed with the raw odometry data and (b) computed by efficient FastSLAM algorithm in real-time with 100 particles. Reproduced from Hahel et al. (2003).



(a) (b)

Figure 2.8: Robot trajectories of all 100 particles (a) shortly before loop closing and (b) after loop closing. Reproduced from Hahel et al. (2003).

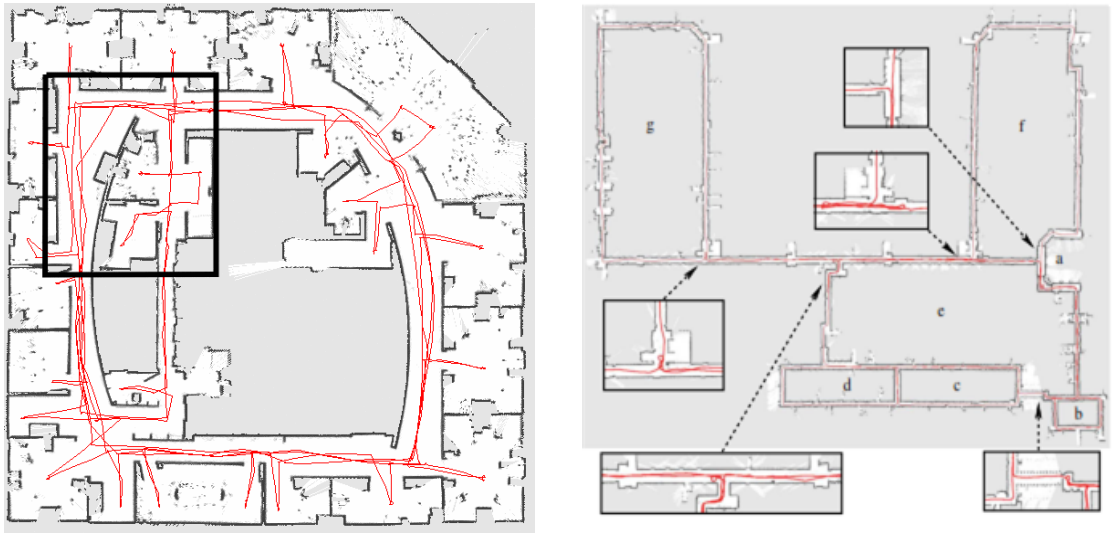
2.2.2 GMapping Algorithm

Similar to the efficient FastSLAM algorithm, Grisetti, Stachniss, and Burgard (2007) proposed gmapping, another efficient particle filter based approach. It also uses scan-matching techniques (Montemerlo et al., 2002) but this technique uses all observation data and odometer data (including the most recent one) to estimate the robot's position more accurately. Consequently, this algorithm further reduces the number of particles without compromising map quality (for details see table 2 in (Grisetti et al., 2007)). It is improved further by making the resampling step adaptive. The resampling step is carried out only when the total estimation error of all particles crosses a threshold. This procedure significantly reduces the number of resampling steps. As a result, the mapping algorithm decreases computational time as well as the risk of the particle depletion problem.

GMapping algorithm has been tested on the same Intel Research Lab (28m×28m) dataset as the efficient FastSLAM algorithm. Figure 2.9a shows the map computed by gmapping algorithm with 15 particles, whereas, figure 2.7b shows a map computed by the efficient FastSLAM algorithm with 100 particles. Figure 2.9b shows the map for even larger environment (250m×215m).

2.2.3 Graph-Based Algorithm

In a graph-based formulation of the SLAM problem, each node represents a robot position. Observation data (laser scans) taken from that position and the edge between two nodes represents translation and rotation between them. The goal of the graph-based algorithm is to find the best possible position of the robot that minimizes the misalignment between all observations. Grisetti, Kmmerle, Stachniss, and



(a) (b)

Figure 2.9: Map computed by gmapping algorithm for the (a) Intel Research Lab (28m×28m) dataset with 15 particles and (b) The MIT Kilian Court (250m×215m) dataset with 60 particles. Reproduced from Gresetti et al. (2007).

Burgard (2010) proposed an algorithm which minimizes error by using least-square optimization techniques. The algorithm computes a map in the following four steps:

Step1: Get input - Whenever the robot moves more than 0.5 meters or rotates more than 0.5 radians, the algorithm adds a new node to the graph and labels it with the current observation data.

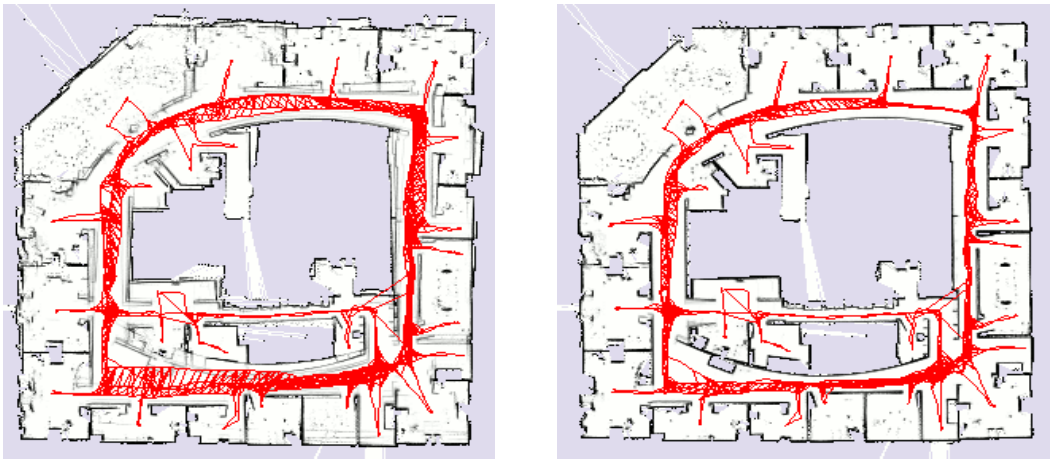
Step2: Data association - For every new node, the new observation data is matched with the previously acquired one by using a scan-matching technique (Olson, 2006) to improve the estimation of the robot position and the corresponding edge is added to the graph.

Step3: Loop closing - When the robot revisits a known area after traveling for a long time, the algorithm looks for matches of the current observation data with the past measurements. If the scan-matcher reports a success, the algorithm adds a new

edge to the graph. The edge is labelled with the relative transformation that makes the two observations to overlap best.

Step4: Map optimization - The algorithm performs the optimization step whenever a loop closure is detected.

Figure 2.10 shows the map computed by using graph-based algorithm. This algorithm provides a more accurate map with less computational time compared to a filter based algorithm (Burgard et al., 2009). Consequently, in the SLAM community, graph-based algorithms are considered as the state-of-the-art techniques to compute accurate map efficiently.



(a) (b)

Figure 2.10: Intel Research Lab, (a) before optimizing robot position overlayed on top of the resulting map and (b) after optimizing robot position and the resulting consistent map. Reproduced from Gresetti et al. (2010).

2.2.4 Conclusion

The implementation of the basic idea of computing a map from integrating successive views using robots reveals that this process is not a straightforward one. It requires

one to continuously track one's movement in space and to constantly maintain all possible maps until later information can help to decide which one is the best representation to use. However, humans do not maintain a precise map in their head and have little interest in localizing their exact position for every step they take. They might jump, skip, and roll for fun. As such, how do they compute a map of their environment and what kind of a map is computed?

2.3 Cognitively-inspired Mapping

The use of robots for mapping studies is inextricably linked. As robots explore their environment, they need to learn what and where things are; in short, they need a map. Various approaches have been developed in the past using robots to study the mapping problem. Some have simply found a working solution for the robots while others have attempted to create a more cognitive-like solution. The former, focusing on efficiency and reliability, is often referred to as robot mapping. The latter, focusing on including some aspects of cognitive mapping, is often referred to as cognitively inspired mapping (for examples of such works, see Jefferies and Yeap (2008)).

Cognitively inspired mapping takes several forms. Some robots are created with physical properties (e.g. sensors) that allow them to be functionally equivalent to animals or insects (Lambrinos, Mller, Labhart, Pfeifer, & Wehner, 2000) while other research focuses on other similarities such as the creation of robots with a neural brain (Milford & Wyeth, 2007; Barrera & Weitzenfeld, 2008; Hafner, 2008) or bio-inspired behavior (Chown, Kaplan, & Kortenkamp, 1995; Jefferies, Baker, & Weng, 2008; Beeson, Modayil, & Kuipers, 2010). Although these approaches are different,

their objective is similar, to compute a map of the environment. In the following, we review three different approaches and the approach adopted in this study, the Albot approach.

2.3.1 Sahabot 2

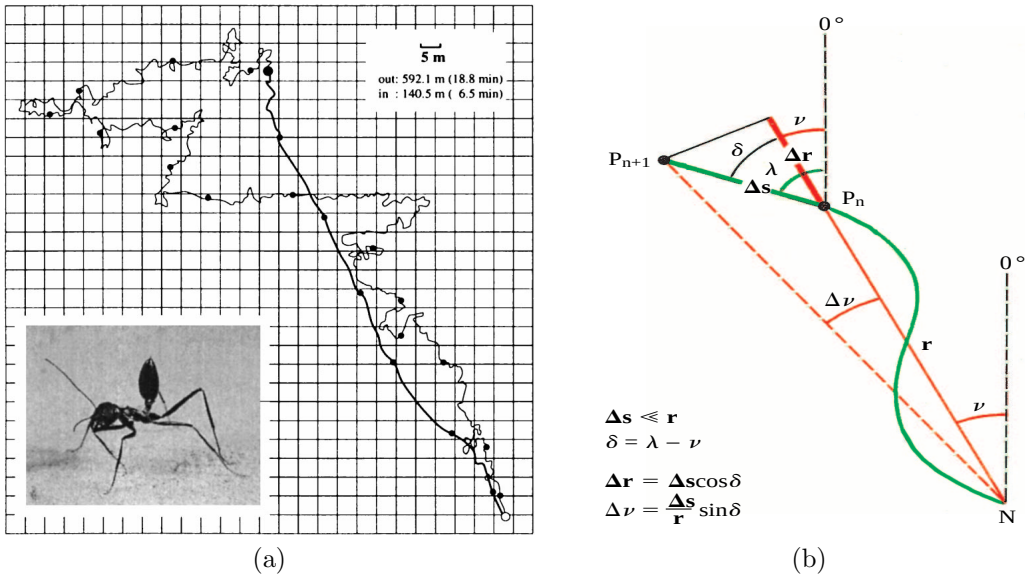
Lambrinos et al. (2000) created Sahabot 2 (a robot) that perceives and finds its way like a desert ant *Cataglyphis*. They created Sahabot 2 based on Wehner, Michel and Antonsen's (1996) findings of *Cataglyphis*'s navigation strategies. Wehner et al. found that *Cataglyphis* travels in a circuitous path towards food source, but comes back to their nest, following the almost straight path as shown in figure 2.11a. They do so by means of three navigational strategies such as path-integration, landmark-based visual homing, and systematic search.

Path integration - the primary mechanism in *Cataglyphis*'s navigation system, updates its nest location and direction with respect to the current position by continuously adding all angles turned and all distance traveled as shown in figure 2.11b. To find turned angle, they use a polarization pattern in the skylight as a compass. Detecting the level of polarization at their facing direction, they find the location of the sun and calculate an angle. To find the travel distance, they use optical flow (change of position of objects, surfaces, etc. in a visual scene due to the relative motion between an observer and the scene (Raudies, 2013)) in their retinal image.

Visual homing - the secondary mechanism in *Cataglyphis*'s navigation system used to find the exact location of their nest. As the path integration process is inherently prone to cumulative errors, they can only return to the vicinity of their nest. At that point, the landmark based navigation system takes place to find the

direction required to reach the nest by comparing snapshots taken at the nest location and current location.

Systematic search - an alternative mechanism of visual homing in *Cataglyphis*'s navigation system. They use a "non-random systematic search" technique to reach the nest in the absence of apparent landmarks near their nest.

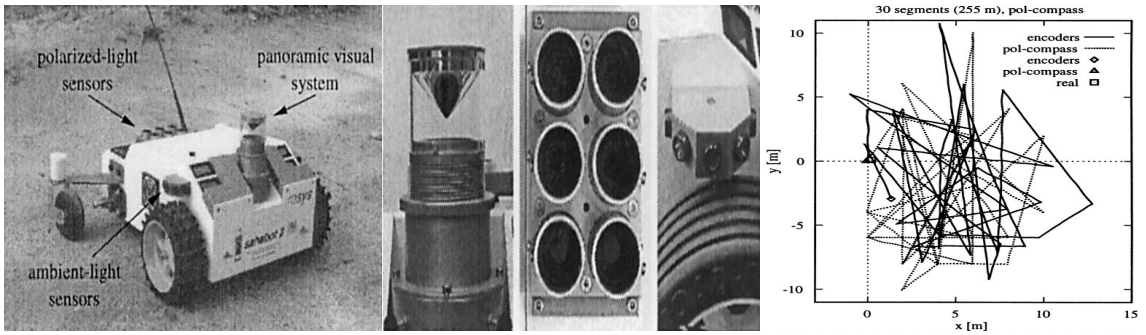


(a) (b)

Figure 2.11: (a) A typical foraging trip of the Saharan ant *Cataglyphis*(inset). Starting at the nest (open circle), the ant searches for food on a random course (thin line) until it finds a prey (position marked with the large filled circle). The food is carried back to the nest on an almost straight path (thick line). Reproduced from Wehner and Wehner (1990). (b) Path-integration scheme. The position P_n of the animal in relation to the starting point of its foraging excursion (nest) is described by the vector $(\upsilon, \mathbf{r})$. If the animal proceeds in the direction λ by a path increment Δs, the new position $P_{n+1}(\upsilon + \Delta\upsilon, \mathbf{r} + \Delta\mathbf{r})$ is reached. Reproduced from Hartmann and Wehner (1995).

Lambrinos et al. (2000) employed a polarized-light sensor and an ambient-light sensor as shown in figure 2.12a-b. These two sensors find orientation with respect to the solar meridian (line passing through the sun and the zenith). Initially, the

polarized-light sensor provides two possible orientations (θ and $\theta + \pi$) by detecting polarized patterns in the skylight. Later, an ambient-light sensor provides correct orientation based on the stronger response from both orientations. To find the traveled distance, they used directional information from the previous step along with the velocity of the robot measured by the wheel encoders. Figure 2.12c shows the result of Sahabot's path integration process compared to a traditional proprioceptive system for 255 meters traveled, when an error between the real position and the estimated position is 29 cm. To find the precise location of the nest, they proposed an average landmark vector model based navigation system using panoramic visual information as shown in figure 2.13a. This process finds the direction required to reach the goal by comparing panoramic images taken from the nest and current position. Figure 2.13b shows an example of an experimental setup where four landmarks are placed in the desert to test the navigation system. Figure 2.13c shows navigation performance in a simulated environment with 27 landmarks.



(a) (b) (c)

Figure 2.12: (a) The mobile robot Sahabot 2. (b) Robot sensors. From left to right: the panaromic visual system, polarized-light sensors, ambient-light sensors. (c) Trajectories estimated by both the POL-compass (Sahabot's polarized compass) system and the proprioceptive system during an experiment with a 30-segment trajectory. The final positions of the robot as estimated by both systems and the real final position are also drawn. Reproduced from Lambrinos et al. (2000).

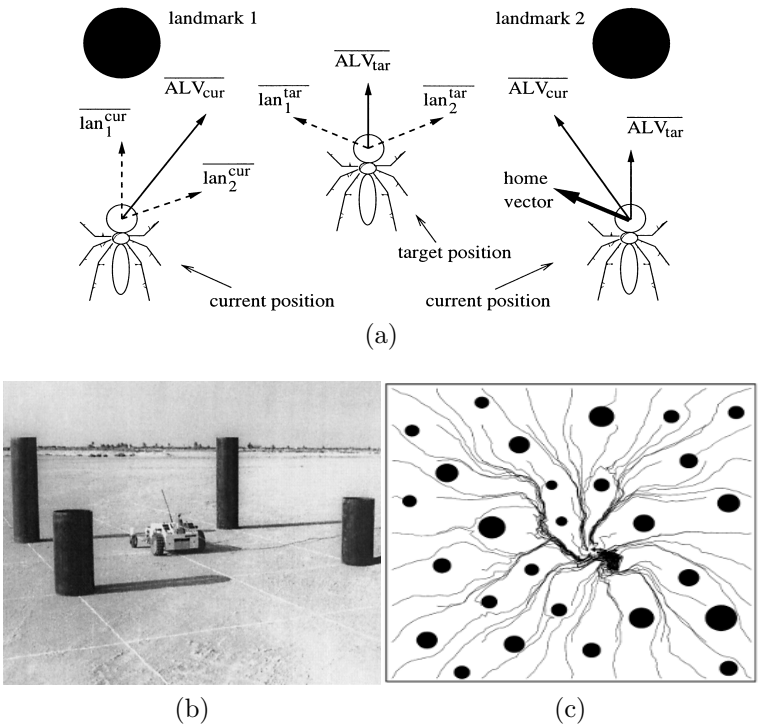


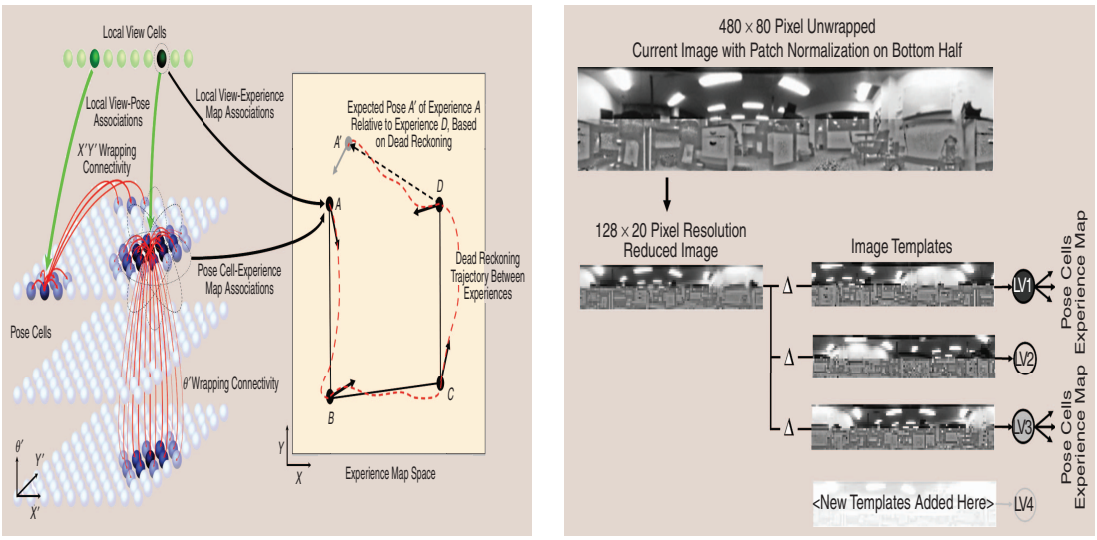
Figure 2.13: (a) Diagrammatic description of the Average Landmark Vector (ALV) model. Each landmark feature i in the visual field - in this version of the model sector centers - is associated with a unit vector, called landmark vector ($\overline{lan_i^{cur}}, \overline{lan_i^{tar}}$). All landmark vectors are averaged to produce a single Average Landmark (AL) vector. At every point, the home vector - computed as the difference between the current AL vector and the vector at the target position ($\overline{ALV_{cur}}, \overline{ALV_{tar}}$) - gives the approximate direction to the target position. (b) Example of a landmark array used for the navigation experiments. The grid visible on the desert ground was used for the alignment of landmarks and robot, and for the registration of the final robot position. (c) The performance of the homing models in a complex environment with 27 landmarks. Reproduced from Lambrinos et al. (2000).

2.3.2 RatSLAM

Wyeth and Milford (2009) proposed the RatSLAM model for practical robot navigation based on neurological findings of the rat's spatial mapping. In particular,

this work is inspired by the rat's strong navigational ability, as they can navigate in featureless large-scale environments (e.g. narrow tunnel) with their noisy odometer.

The RatSLAM model can be summarized as follows. It consists of three components: pose cell, local view cell, and experience map to emulate rats mapping and navigation mechanism. The pose cells form a three-dimensional network as shown in figure 2.14a. Each unit in the network corresponds to a specific robot's position and orientation in the environment. A group of neighbor's pose cells corresponds to a specific location in the environment represented by a local view cell. Each local view cell contains a panoramic image template captured by a camera and parabolic mirror as shown in figure 2.14b. Image templates provide a unique signature for lo-



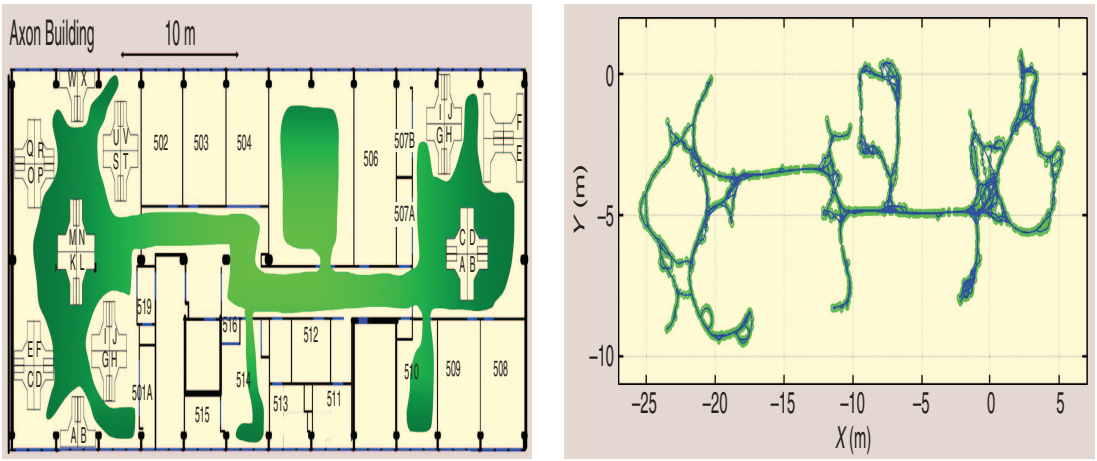
(a) (b)

Figure 2.14: (a) The RatSLAM model. Pose cells forms a 3-D networks, each pose cell represents the robot's position. Each local view cell represents a specific location in the experience map. (b) Visual interface to the local view cells. Reproduced from Wyeth and Milford (2009).

cations. Using a SLAM based approach; a two-dimensional global map is computed

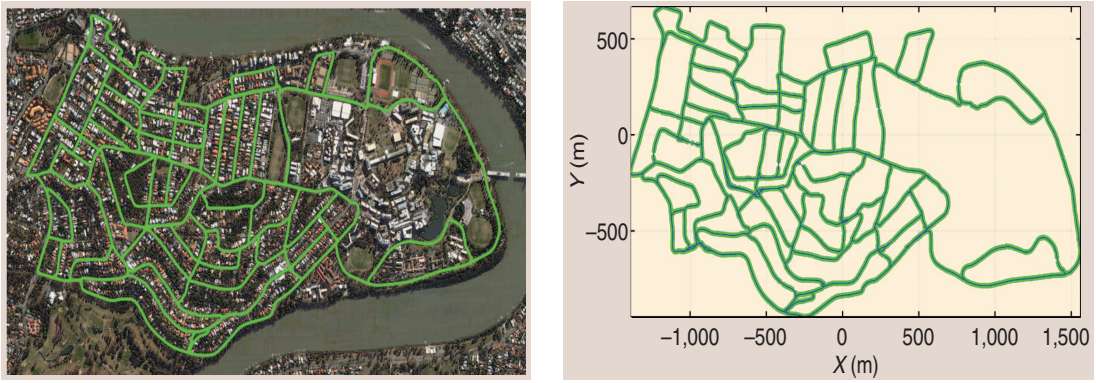
to represent the environment visited by the robot. On the global map, each node is associated with a local view cell i.e. a location. A connection between two nodes represents the distances between them. When the robot moves through the environment and captures a new image, it compares it with all template images. If results go below a threshold, it adds a new view cell which represents a new location in the map. During a loop closing event, when it finds few consecutive new images and template images (from memory) are matched, it corrects position in the map.

The RatSLAM algorithm has been tested on an autonomous delivery robot in large-scale indoor (figure 2.15) as well as outdoor (figure 2.16) environments. Unlike traditional SLAM based algorithms, RatSLAM does not compute a map with geometric features. Instead, it only computes a route map. During place re-visiting, it uses image features to recognize places and close loops. Afterwards, it uses loop closing information to compute the precise map.



(a) (b)

Figure 2.15: (a) An indoor office like environment and (b) computed route map. Reproduced from Wyeth and Milford (2009).



(a) (b)

Figure 2.16: (a)An outdoor environment and (b) computed route map. Reproduced from Wyeth and Milford (2009).

2.3.3 Hybrid Spatial Semantic Hierarchy (HSSH) Model

Kuipers (1978) suggested that humans extract and process spatial information in different interconnected forms such as route description, topological network, boundaries of place. From this theory, he developed a simulation model named TOUR. Later, he extended that idea and proposed the spatial semantic hierarchy (SSH) model (Kuipers, 2000; Remolina & Kuipers, 2004). Recently, Beeson, Modayil, and Kuipers (2010) extended the SSH model further and proposed the hybrid spatial semantic hierarchy (HSSH) model. In the HSSH model, knowledge from sensor level is represented in four levels in a hierarchical manner as shown in figure 2.17.

The four levels of representation can be summarized as follows:

1. **Local Metrical Level:** At this level, the robot computes a precise metrical description of its surrounding space which is referred to as local perceptual map (LPM). Like many SLAM based algorithms, the LPM is computed as an occupancy grid representation (Moravec, 1988) using particle filter based

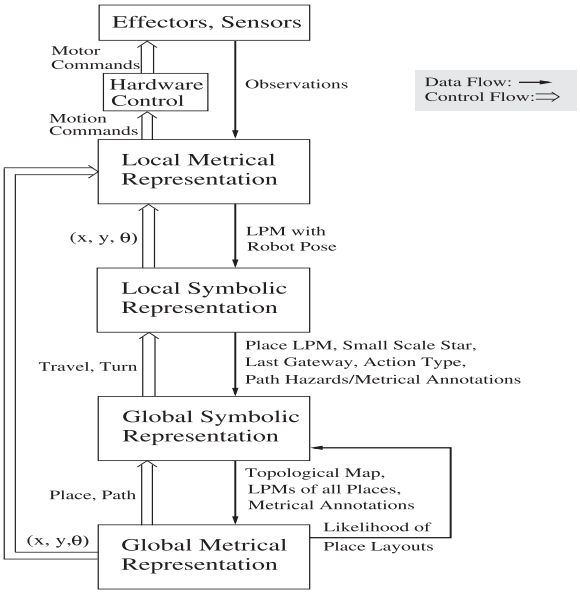


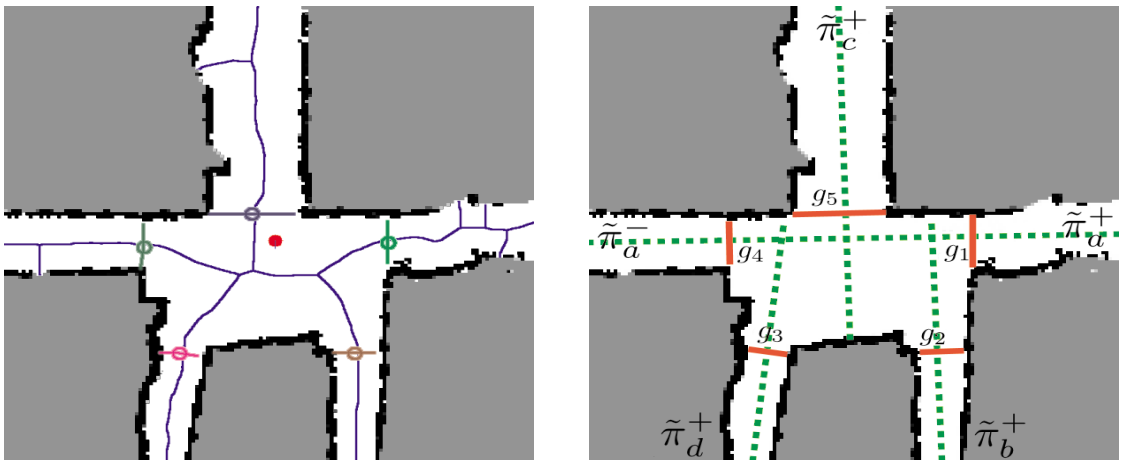
Figure 2.17: HSSH description. Reproduced from Beeson et al. (2010).

localization techniques (Fox, Burgard, & Thrun, 1999). The LMP represents the small scale-space with a bounded size in a local frame of reference by correcting noisy odometer data and integrating observation data over time. The LMP is to provide information for local motion planning and obstacle avoidance.

2. **Local Topological Level:** When the robot moves through the environment, it identifies discrete places (e.g., corridor intersections, rooms, etc.) in the large-scale continuous environment and finds corresponding local topologies. A local topology is a set of directed local-paths in a circular order and each local-path is associated with at least one gateway. Directed local-paths are the pruned Voronoi graph skeleton of LMP which provides information to reach neighbor places, whereas gateways are the gaps in LPM boundary which separates local place from the larger continuous environment as shown in figure 2.18.

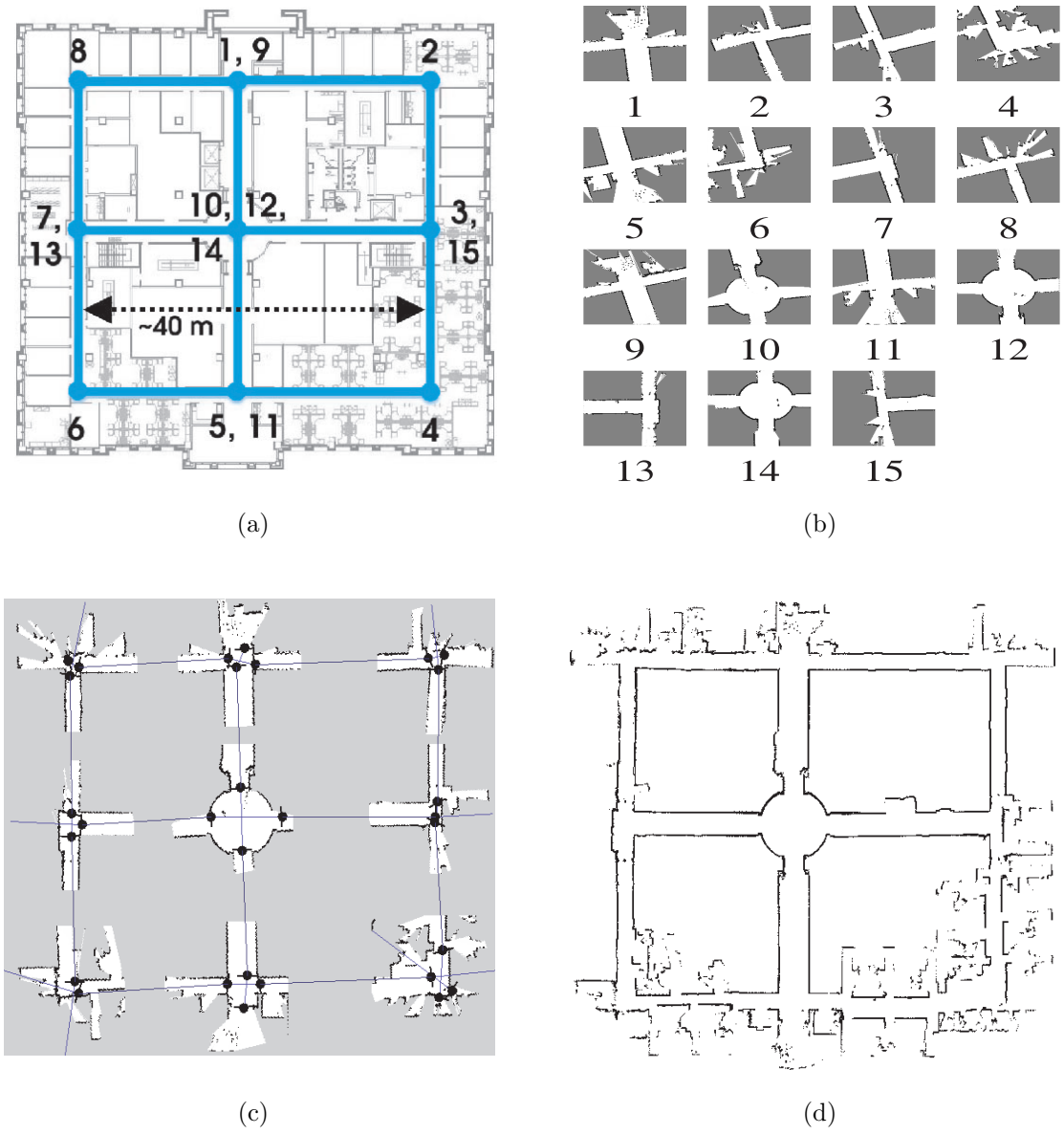
3. **Global Topological Level:** At the global level, the robot represents the entire environment as a graph of places and paths connecting them. This representation is referred to as a global topological map. When the robot moves into a new place, either it creates a new place in the graph or joins with the previous place (loop closing) by matching local topology and LPM. Figure 2.19a-c shows a test environment, different LPMs, and a global topological map overlaid with the LPMs.

4. **Global Metrical Level:** At the final level, the robot computes a global metrical map which describes the layout of places, paths, and obstacles within a single global frame of reference. It computes such a map by combining all places using the skeleton provided by the topological map. Figure 2.19d shows the resultant global metrical map of the environment shown in figure 2.19a.



(a) (b)

Figure 2.18: (a) A local perceptual map (LPM) with pruned Voronoi skeleton (dipected by blue lines) and Gateways (dipected by line passing through circles) and (b) a LPM with local topological information (directed local paths and gateways). Reproduced from Beeson et al. (2010).



(a) (b)

(c) (d)

Figure 2.19: (a) A test environment. (b) The LMPs produced at each place, (c) the global topological map with overlaided LPMs after matching local topologies and LMPs, and (d) the final global metric map. Reproduced from Beeson et al. (2010).

2.3.4 The Albots Approach

The Albots approach is unique in that it does not implement some known aspects of spatial behavior onto a robot. Rather, it identifies a problem confounding cognitive researchers interested in spatial cognition. It then attempts to find an answer to the problem via the implementation of a robotic system that can solve a similar problem in its own environment and with its own embodiment. One example is the problem tackled in this thesis: how do we compute an incomplete and imprecise map from successive views of the environment? We implement a solution using a robot equipped with a laser sensor. Another possible solution might utilize the recognition of landmarks. Many cognitive researchers note the importance of landmarks in learning the environment and identify several factors affecting our perception of them. Yet, it is unclear why some information is remembered as landmarks and other information is not, and why only a few landmarks are remembered when learning the environment.

Albots are created to provide insights into possible solutions to these hard problems. As such, one is not looking for a solution that is, say, more efficient than the robot mapping approach, but rather one which is more consistent with findings from studies of cognitive systems. As Yeap (2011b) noted, this is the essence of the Albot approach and its significance thus lies in its explanatory power rather than its demonstrative power. However, this approach complements the other two approaches (see figure 2.20). It complements the robot mapping approach in discovering algorithms as well as the cognitively inspired approach in discovering ideas.

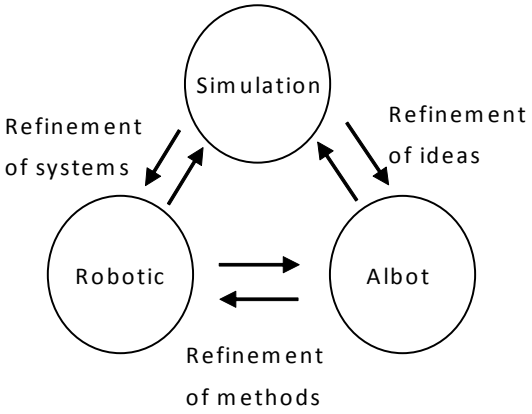


Figure 2.20: Cross-fertilization of ideas in mapping research. Reproduced from Yeap (2011b).

Albot$_0$

Albot$_0$ is the first of its kind (Yeap, 2011b; Wong, 2008). It was created to investigate what it means to have an inexact map and in particular, whereby one cannot read the exact location of where one is from the map. With such a map, how does one find one's way home? To create Albot$_0$, a robot equipped with sonar sensors and an odometer was chosen. Both of its sensors are known to be highly inaccurate and without any error correction routine, the shape of the room computed as Albot$_0$ moves from one end to the other is highly imprecise. It was later discovered that these local spaces computed during the outward journey are different from those computed during the inward journeys. The map computed during the outward journey provides a description of the environment based on the outward perspective and in the inward journey, one computes a map based on the inward perspective. Note that the latter was not used to update the former because the shapes of the local spaces computed are imprecise and no matching could be done. As such and with very limited sensing, how could Albot$_0$ find its way home?

Inspired by how animals often rely on the distance traversed and orientation in the local environments, Albot$_0$ finds its way home by computing the distance traversed between exits in each local space visited. Even though the number of local spaces computed in the outward and inward journeys is different, the sum of the distance traversed is almost the same. This crude method allows Albot$_0$ to find its way home most of the time. One trial of the experiments conducted is shown in figure 2.21.

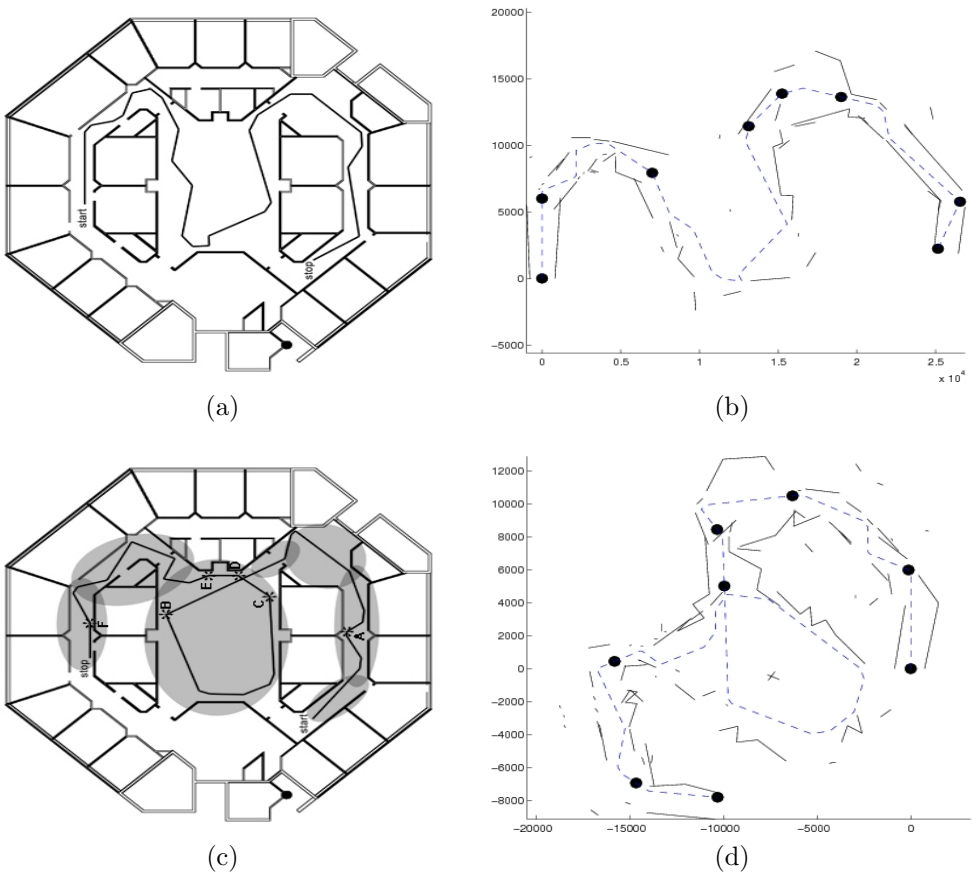


Figure 2.21: (a) Test environment with Albot$_0$'s trajectory during exploration and (b) map computed. (c) Test environment with Albot$_0$'s trajectory during returning home and (d) map computed. Reproduced from Yeap (2011) and Wong (2009)

During this trial, as shown in figure 2.21a, Albot_0 traveled from a location indicated by "start" and stopped at the location indicated by "stop". For this outward journey, it computed seven local spaces as shown in figure 2.21c by the oval and circular shades. In figure 2.21b and d, a local space is represented as the area between two successive filled circles. Figure 2.21b and d show that the maps of a local space computed in the outward and inward journeys are different. Even though the shape of the the local spaces in the outward and inward journeys are different, Albot_0 was able to reach home. Interestingly, Albot_0's success rate of returning home was 70%, making it a type of creature who fails to return home sometimes.

2.4 Conclusion

In this chapter, we reviewed three psychological experiments that investigated what kind of a map is computed in our head. The surprising models developed suggest that the mapping process is not as straightforward as integrating successive views to form a map. We also reviewed three best-known SLAM algorithms and show that these algorithms work by focusing on tracking robot position for all steps and correcting sensor errors and produce precise maps. We also reviewed three cognitively inspired robot mapping algorithm. Like SLAM approaches, these algorithms also produce an accurate map by tracking every robot step precisely. In the next chapter, we show how an alternative process can be implemented on Albot_1.

Chapter 3

On Designing and Implementing Albot$_1$

Albot$_1$ is a robot equipped with a laser ranging device and an odometer; embedded with an algorithm for computing a map of its environment based on Yeap's (2011) theory of perceptual mapping. In this chapter, we describe the implementation of Albot$_1$ and six different experiments conducted to evaluate Albot$_1$'s performance. These experiments concern: (i) Albot$_1$'s ability to compute a map in an office-like environment, (ii) Albot$_1$'s ability to handle errors, (iii) Albot$_1$'s ability to orient itself, (iv) Albot$_1$'s ability to compute its map using continuous motion, (v) and (vi) Albot$_1$'s performance compared with SLAM-based algorithms. The success of experiment 1 meant that Albot$_1$ is able to compute an inexact and incomplete map for its environment. The remaining experiments are designed to investigate the robustness of this new algorithm.

Section 3.1 describes the implementation of Albot$_1$'s perceptual mapping, from a flow chart of its process to the robot used and the test environment. Section 3.2 describes the six experiments and results obtained. Section 3.3 concludes this chapter with a discussion of the lessons learned.

3.1 Albot$_1$'s Perceptual Mapping Algorithm

Albot$_1$'s perceptual process according to Yeap's theory can be described as follows:

1. Each time Albot$_1$'s perceptual map is updated with a view, Albot$_1$ is entering a new local environment defined by the view.

2. As Albot$_1$ explores its local environment, it keeps track of landmark objects.

3. Just before all these landmark objects disappear from view, Albot$_1$ triangulates its position in its perceptual map and adds a new view to the map.

4. During updating, information from the incoming view over-writes any information in the map that occupies the same space.

5. The process repeats itself.

The theory leaves open four implementation issues: 1. What objects/surfaces are selected as landmarks. 2. How such landmarks are recognized in successive views. 3. When exactly should a new view be added. 4. How information in that incoming view over-writes information occupying the same space in the perceptual map.

Given that Albot$_1$ has only a laser-ranging device to sense its environment, its ability to detect landmarks is limited to perceived surfaces in view, each being represented as a 2D line. Furthermore, since these lines are recovered from laser points, their shape varies between views depending on the robot's vantage point. Consequently, surfaces with a minimum length of 40cm that have an occluding edge are chosen as landmark surfaces. Having an occluding edge ensures a reference point exists on the surface for triangulating the position of Albot$_1$ and the position of the new surfaces in the map. A minimum of two such points are needed.

Albot$_1$ cannot recognize these landmark surfaces directly from one view to the next. This has to be done indirectly, via the traditional process of transforming information in one view to the next. Transforming between two successive views, the robot can predict where these landmark surfaces will be. They are then "recognized" via the use of some heuristics such as the proximity and orientation of two surfaces, the sudden appearance of a new surface in front of another, and others. It is emphasized that the transformation method is used here because Albot$_1$ lacks the ability to recognize objects in view.

Albot$_1$ must update its perceptual map prior to all landmark surfaces disappearing from the current view. We refer to this point as the limiting point in the current space and we define it as a view that contains less than three landmark surfaces. However, note that if Albot$_1$ reaches the limiting point of its current local environment, it would not be able to update its map due to insufficient points to triangulate its position in the map. As such, when Albot$_1$ reaches its limiting point, it will update its map using its previous view. During updating, the robot's current position in the map is triangulated using one of the recognized landmarks of previous view, and the view is transformed and added to the map.

When a new view is added to the map, its space could be occupied by some other information. Due to errors, the latter may or may not be the same information. Nevertheless, the information is deleted so that this part of the map contains only the fresh information derived from the current view.

Figure 3.1 shows the stages involved in the implementation of Albot$_1$'s perceptual mapping process. The remainder of this section describes these steps in detail.

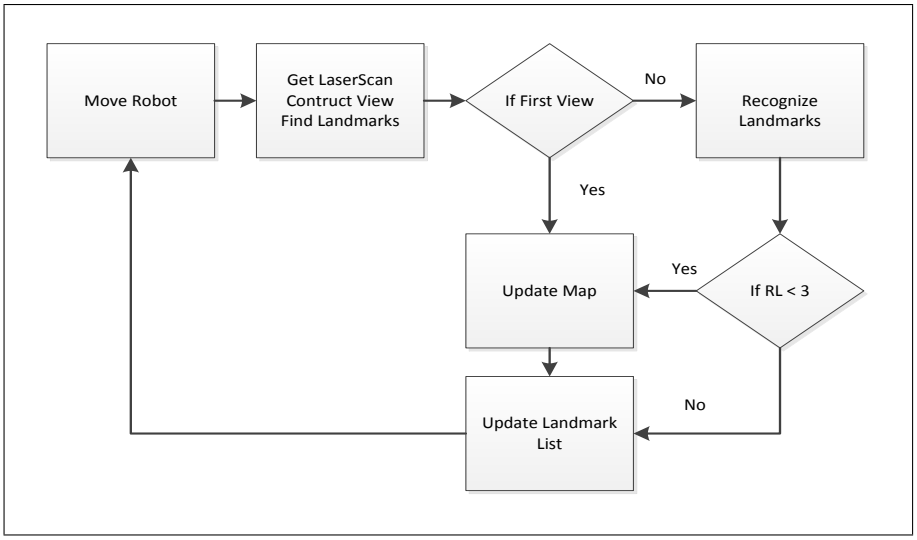


Figure 3.1: Flow chart of Albot$_1$'s perceptual mapping process.

3.1.1 Experimental Platform: Mobile Robot and Sensors

The base Pioneer 3 DX robot platform (figure 3.2) from MobileRobots Inc. has been used for this research. It is composed of an embedded controller, motors with 500-tick encoders (odometer), 19cm wheels, and an aluminum body. It has two solid rubber tires, driven by a two-wheel differential, reversible drive system. A caster wheel is positioned at the rear of the robot to provide balance. It can reach speeds of 1.6 meters per second and carry a payload of up to 23 kg. It is powered by three hot-swappable 9Ah sealed batteries. When the robot moves from one position to another, an odometer provides its absolute position (x, y, theta) relative to starting position.

The robot platform is equipped with a Laser Measurement Sensor (LMS) 200 (mounted on top of the robot) to perceive its environment. This LMS-200 is a 2-D laser scanner manufactured by SICK optics. It has a class 1 infrared laser and operates with the principle of time-of-flight. It can provide distances to objects over 180 degree area with 0.25, 0.5 or 1.0 degree angular resolution up to 80 meters away.

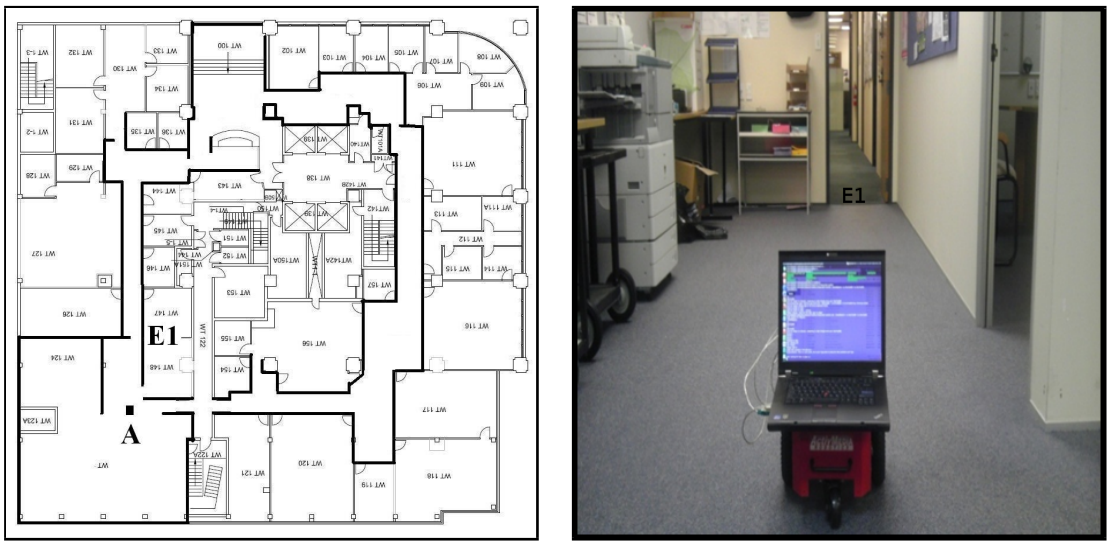
A laptop computer is placed on top of the robot for processing information provided by the robot's odometer and laser sensor. This laptop is installed with Pioneer SDK (C++ library) and connected to the robot's embedded controller and laser sensor via USB-to-Serial connector. Higher level control programs created with the Pioneer SDK run on laptop computer to control the robot and gather distance to objects as well as the robot's absolute position in a 2-D coordinate system.

Figure 3.2: Robot platform (Albot_1)

3.1.2 Test Environments and Inputs

Albot_1 was tested in a standard office-like environment with a carpeted floor. Figure 3.3a shows the floor plan of the test environment. It should be noted that there are different objects (chair, cupboard, rubbish bin, etc.) throughout the whole environment. Therefore, layout used in this thesis does not represent the exact environment. It has been used to show the basic shape of the test environment. Figure 3.3b shows a picture of the actual environment in which Albot_1 "lives". Albot_1 in figure 3.3b is

at point A in figure 3.3a. Exit E1 on both figures represents the same door in the environment. During tests, Albot$_1$ is guided through the environment in two different



(a) (b)

Figure 3.3: (a) Test environment (Floor plan of level one, AUT Tower). (b) A snapshot of real environment while the Albot$_1$ is scanning for inputs.

manners. When it is guided in a start-stop manner, it takes a laser scan after every step (usually 1m or 45 degree) it completes. When it is guided in a continuous manner, it takes a laser scan after every certain translation (10 cm or 1m) or rotation (1 or 10 degree). During scanning, if 1 degree angular resolution is set, the laser scanner first emits 181 laser beams over a 180 degree area and receives reflected beams. Depending on their time of flight and angle, locations are found for all reflecting points within 180 areas. As a result, initial laser data produced by these laser scans are usually in the form of $P = \{(x_i, y_i), i = 1, 2, \ldots, n_p\}$, where (x_i, y_i) is the Cartesian coordinate of the i-th reflected point and n_p (180 for 1 degree angular resolution) is the total number of reflected points of each scan. Consequently, an initial laser scan

contains a set of 2d coordinates of reflecting point's coordinates over a 180 degree area where Albot_1's position is at the origin facing positive y-axis.

Figure 3.4 shows a typical laser scan where green lines represent laser beams scattered over a 180 degree area, red dots represent reflecting point, and the center red square represents Albot_1's position, the blue line on the red square represents Albot_1's facing direction. In fact, Albot_1's position and facing direction can be represented by a single line (position represented by first edge point and facing direction represented by direction from the first edge point to the last point). In this thesis, a square around the first edge point is drawn for better representation.

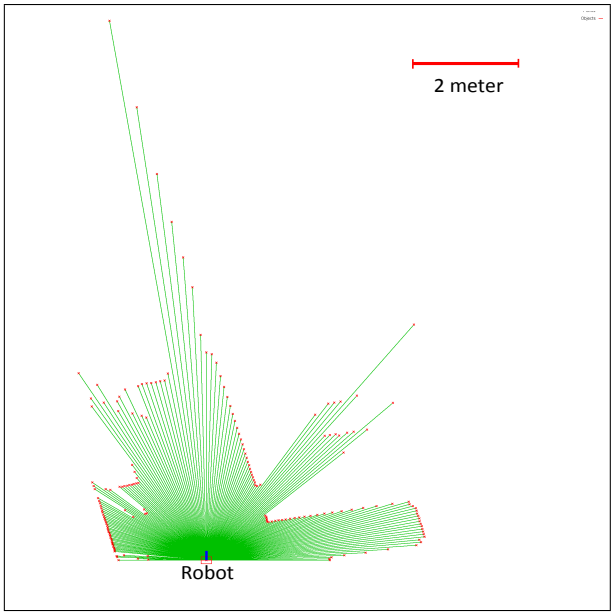


Figure 3.4: An example laser scan (robot position is same as shown in figure 3.3b)

3.1.3 Constructing View

After getting an initial laser scan, the next step in the algorithm is to construct a view by extracting surface (line) features from 2D point clouds. Several well-known

algorithms are available to extract lines from a laser scan. They include popular split-and-merge algorithm (Borges & Aldon, 2004), robust RANSAC Random Sample Consensus algorithm (Fischler & Bolles, 1981), simple incremental algorithm (Vandorpe, Brussel, & Xu, 1996), and probabilistic expectation-maximization algorithm (Pfister, Roumeliotis, & Burdick, 2003) All these algorithms are intended to compute an accurate representations of the environment (for an evaluation based on accuracy see (Nguyen, Martinelli, Nicola, & Siegwart, 2005)). However, as the main objective of this thesis is to implement an algorithm to compute an imprecise and incomplete map, we implemented a simple algorithm to construct a surface based view representation. The steps of the view construction algorithm can be described as follows (see figure 3.5 for graphical representation):

1. **Point Filtering:** Among initial laser readings, if there is any point situated beyond a distance threshold (max_range), that point would be deleted from the list.

2. **Point Clustering:** After filtering, all laser readings are clustered based on Euclidean distance between two consecutive points. If this distance exceeds a threshold (cluster_gap) then a new cluster is formed for the second point.

3. **Cluster Segmenting:** For each of the clusters formed, if the cluster size exceeds a threshold (cluster_size) then a further check is done to find whether that cluster needs to break. This is done based on the perpendicular distance from each point to the line joining the first and last point of that cluster. If this distance exceeds a height threshold (break_point_height) then that cluster is broken down into two from that point. Step 3 is repeated for the two newly

formed clusters.

4. **Surface Forming:** For each of the clusters formed, a surface is formed by simply drawing a straight line from the first point to the last point.

5. **Surface Filtering:** All surfaces are further filtered out based on their length. If the length of any surface is less than a threshold (surface_length) then that surface would be deleted from the list.

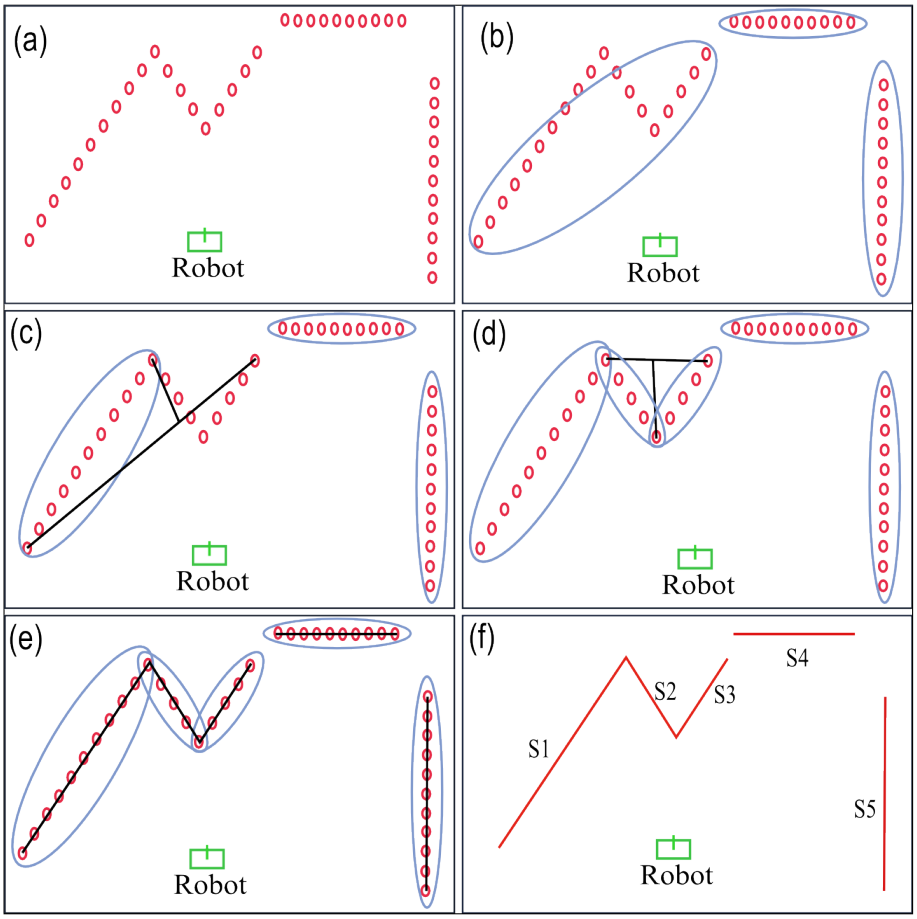
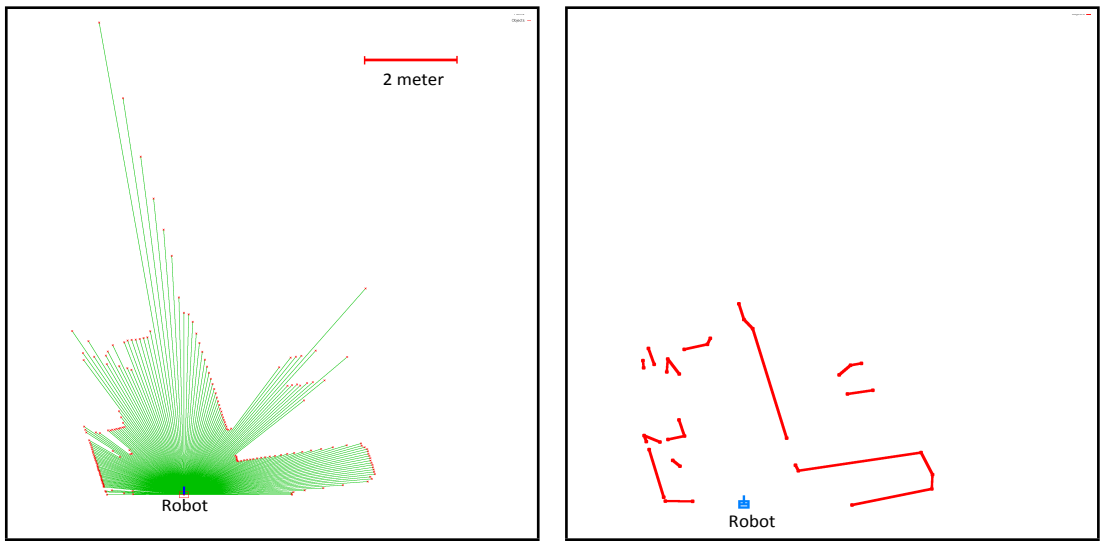


Figure 3.5: A graphical representation of the surface extraction algorithm.

For this research, max_range threshold is set to 30 meters. It is seen that laser readings beyond this value are mostly inaccurate. As the robot platform used is of width 50cm, cluster_gap threshold is set to 60 to ensure the gap is such that the robot can go through. After a few trials and tuning, it is found that 20cm cluster_size, 10cm break_point_height, and 10cm surface_length provides a reasonable representation for an initial laser scan.

Applying the above algorithm, a set of surfaces $\{S_1, S_2, \ldots, S_i\}$ is extracted from the laser readings. Each surface is in the form of $S = \{P_1(x,y), P_2(x,y)\}$, where P_1 and P_2 represents first and last point co-ordinates of that surface. All surfaces extracted from a laser scan constitute a view. Therefore, a view contains a set of surfaces in the form of $V = \{S_1, S_2, \ldots, S_i\}$ where i is the total number of surfaces situated within the robot view. These surfaces are arranged from left to right (clock-wise direction) with the robot at origin (0,0) facing the positive y-axis as shown in figure 3.6.



(a) (b)

Figure 3.6: (a) An example laser scan and (b) corresponding view

3.1.4 Finding Landmark Surfaces

The purpose of the finding landmark surfaces stage is to find landmarks in the view so that Albot_1 can track them and orient itself in the map. The input for this stage is a list of surfaces from an individual view and the output is a list of landmarks chosen based on available line features. However, many surface features such as occluded points, length, and orientation are not useful. For example, occluded points of a surface could change drastically as the robot moves. The orientation of surfaces could vary too much due to the way in which surfaces are constructed from laser points in each view. Consequently, these features cannot be used for localization. However, it is observed that implicit features such as an occluding point (An edge point of a surface which obstructs an edge point of a previous or next surface. For example, in figure 3.7, points p3 and p6 are occluding points p4 and p5 respectively) or corner point (points p1 and p2) appears to be sufficiently stable in subsequent views. It is also observed that the shape of small surfaces (length less than 40cm) are comparatively unstable.

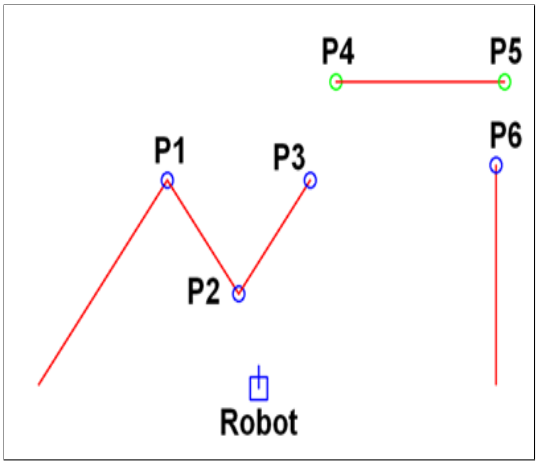


Figure 3.7: A simplified view representing corner points (p1, p2) and occluding points (p3, p6).

Based on the above observations, surfaces with a minimum length of 40cm and which has at least one occluding or corner edge are considered as landmark surfaces. The latter ensures a reference point exists on the surface for relative positioning of Albot$_1$'s location and new surfaces in the map. Figure 3.8 shows landmark surfaces (green lines) for the view shown in figure 3.6.

Figure 3.8: An example view where green lines represent landmark surfaces.

3.1.5 Recognizing Landmark Surfaces

The purpose of the landmark recognition stage is to find and recognize previously seen landmarks in the current view which will be used later to orient Albot$_1$'s position in the map. Inputs for this stage are the lists of landmarks from two subsequent views (the previous and the current view as shown in figure 3.9a-b) in their own coordinate frame and the distance and angle difference of Albot$_1$'s locations from where the two views are captured. To recognize landmarks, as a first step, the current Albot$_1$'s location (x_R, y_R) in previous views is calculated from the known distance and angle

difference by using equations 3.1 and 3.2 (figure 3.9c shows an example of the next robot's location in a coordinate frame of the previous view).

$$x_R = d * \sin(-\theta) \tag{3.1}$$

$$y_R = d * \cos(-\theta) \tag{3.2}$$

$$x_2 = (x_1 - x_R)\ \cos\theta\ +\ (y_1 - y_R)\ \sin\theta \tag{3.3}$$

$$y_2 = (y_1 - y_R)\cos\theta\ -(x_1 - x_R)\ \sin\theta \tag{3.4}$$

where,

(x_R, y_R) is the robots location in a previous view coordinate.

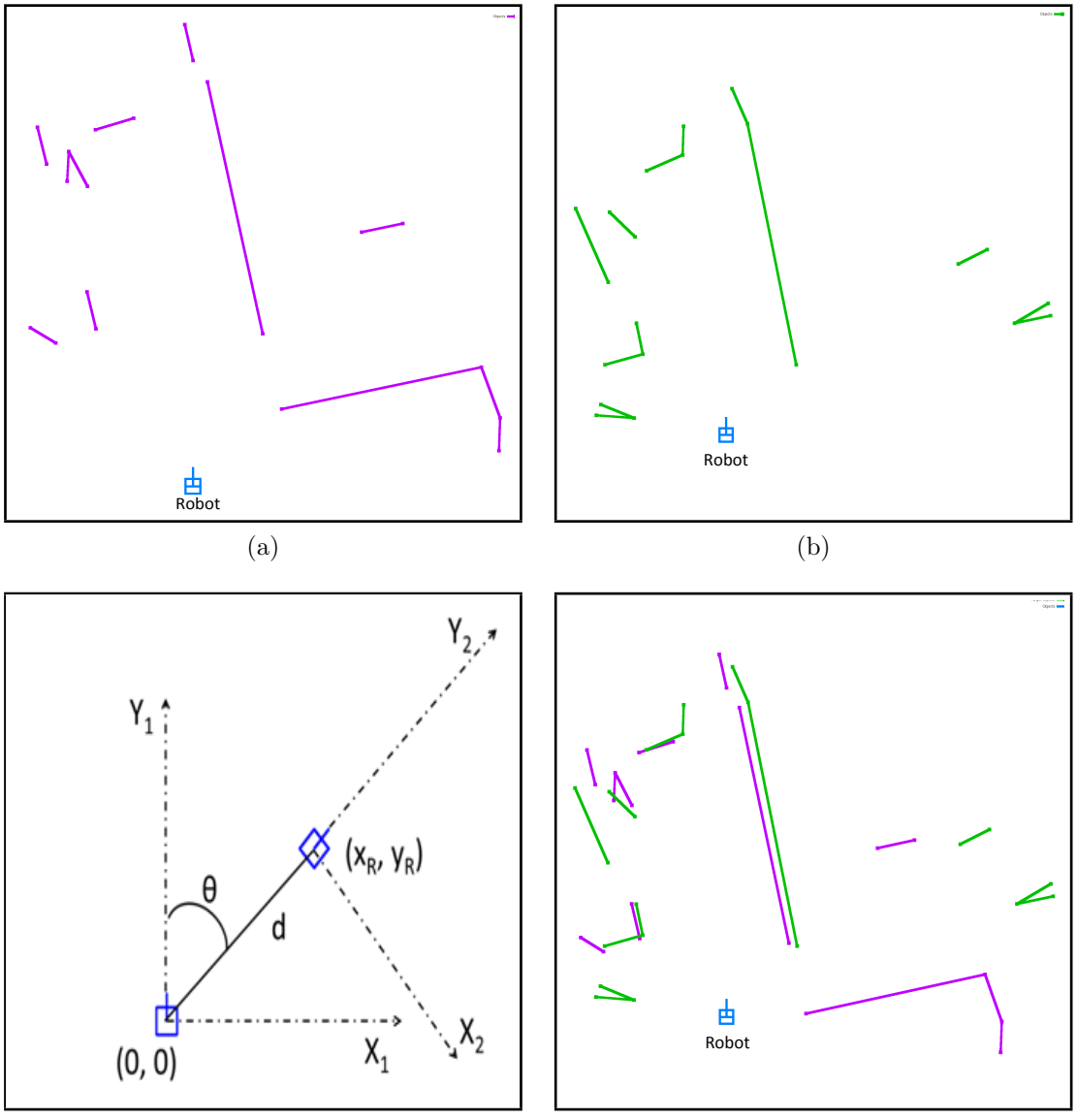
d is the distance between the previous and current robot location.

θ is the angle between the previous and current robot orientation.

(x_1, y_1) is the location of landmarks in the previous view.

(x_2, y_2) is the location of landmarks in the previous view in the current view.

Then, all landmarks in the previous view are transformed onto the current view using equations 3.3 and 3.4. Subsequently, all landmarks from the previous view and the current view are described using the same co-ordinate system as shown in figure 3.9d. In the final step, a landmark pair from the previous view and the current view is considered to be the same if the angle between them is less than 5^0 and the distance between corresponding occluding or corner point is less than 40cm. Algorithm 1 gives a formal description of the algorithm for landmark recognition as described.



(a) (b) (c) (d)

Figure 3.9: (a) Landmarks obtained from the previous view and (b) the current view. (c) The robot's next location in a coordinate frame of the previous view. (d) Landmarks from the previous view after transformation onto the current view.

Algorithm 1 Landmark recognition, recognizeLandmark $(\cdot)$

Input: (i) Recognized landmarks obtained from the previous view, $RL_{n-1} = \{S_1, S_2, \cdots, S_i\}$;

(ii) Landmarks obtained from the current view, $L_n = \{S_1, S_2, ..., S_j\}$; and

(iii) Albot$_1$'s displacement (translation and rotation) from the previous to current robot position, $RD_{TR} = \{\text{distance } (d), \text{angle } (\theta)\}$

Output: A set of recognized landmark surfaces obtained from the current view, $RL_n = \{S_k^n\}$, where k is the number of recognized landmarks.

Begin

for all S_i **do**
    Transform S_i on to the current view by using equations 3.1 to 3.4, $L'_{n-1} \leftarrow S'_i$
end for
for all S'_i **do**
    for all S_j **do**
        if angle $(S'_i, S_j) < 5^\circ$ **then**
            if any occluding/corner point distance $(S'_i, S_j) < 40$ cm **then**
                Replace S_j surface id by S_i surface id
                Save S_j as recognized landmark, $RL \leftarrow \{S_j^n\}$
            end if
        end if
    end for
end for

End

3.1.6 Locating Albot$_1$'s Position in the Perceptual Map

Albot$_1$'s position in its own perceptual map is obtained via triangulation of landmark surfaces. Given that there are usually more than one such surface, the key question is to know which landmark surface to use. Since all we need is an imprecise map, some heuristics rules are used to select landmark surfaces for localization. We introduce two heuristics; one selects the best landmark surface and the other discards those that appear out of line. For example, with three landmark surfaces in view, one

heuristic obtains three localization points for the Albot$_1$'s position in the map. To decide for the worst, we compute the average localization point from these three points and its standard deviation. Any localization point whose distance from the average localization point is greater than the standard deviation is then discarded. The rationale behind the application of this heuristics is that these points should be located close together. To decide which point best represents the robot position, one simple possible heuristic is to choose the landmark closest to the robot. However, we opt to compute a priority value (equation 3.5), we call "goodness", for each of the remaining landmarks.

Each landmark surface has a matching pair of surfaces, S_n in the current view and S_{n-1} in the previous view (these surfaces are present in the map and has same id for retrieval). The goodness value is then calculated as follows:

$$Goodness\ value = \frac{S_L}{d_R \times g_{diff} \times a_{diff}} \tag{3.5}$$

where, d_R is the distance of the occluding point of S_{n-1} from Albot$_1$, g_{diff} is the gap length between the occluding points of S_n and S_{n-1}, a_{diff} is the angular difference between S_n and S_{n-1}, and S_L is the length of the surface S_n. The landmark surface with the highest goodness value is considered as a reference landmark. After retrieving the corresponding reference surface from the map, Albot$_1$'s position is calculated by using equations 3.6 and 3.7 (in this case first point of surface is reference point). Figure 3.10 shows how localization is performed using a single landmark.

$$x^* = x_1 + d_p \times \left[\frac{x_2 - x_1}{S_L} \times \cos\theta - \frac{y_2 - y_1}{S_L} \times \sin\theta \right] \tag{3.6}$$

$$y^* = y_1 + d_p \times \left[\frac{x_2 - x_1}{S_L} \times \sin\theta + \frac{y_2 - y_1}{S_L} \times \cos\theta \right] \tag{3.7}$$

Where,

d_p is the distance between the robot location and the reference point of the current view landmark surface as shown in figure 3.10a.

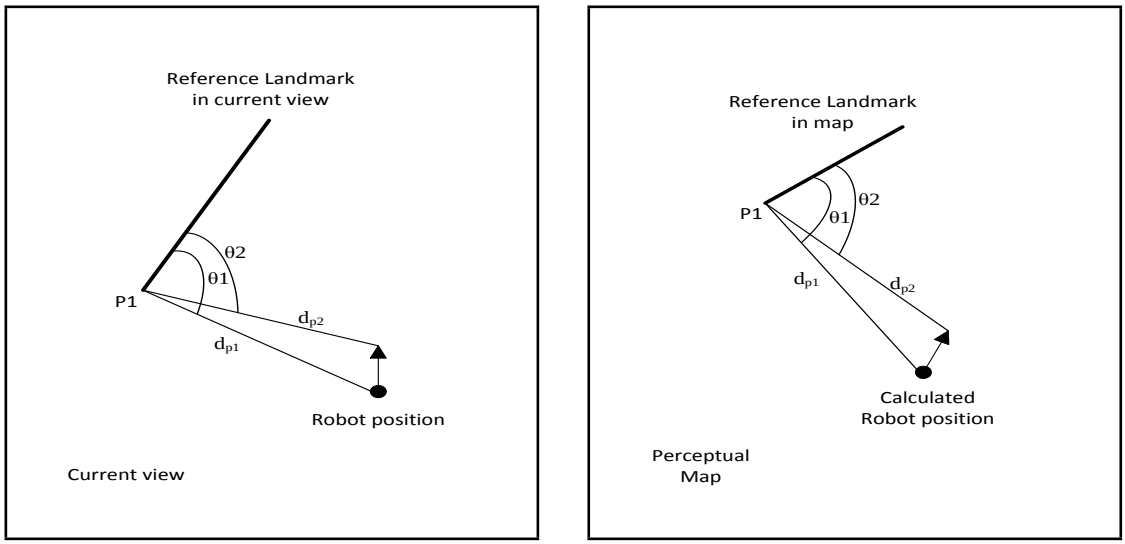
θ is the angle between the current view landmark surface and the line joining the robot location and the reference point of the current view landmark surface.

(x_1, y_1) is the coordinate of first point of the map's landmark surface.

(x_2, y_2) is the coordinate of second point of the map's landmark surface.

S_L is the length of the map's landmark surface.

$(x^*,\ y^*)$ is the location of the robot in the map.



(a) (b)

Figure 3.10: (a) Robot position in the current view co-ordinate. (b) Calculated robot position in the map using equation 6 and 7.

Formally, algorithm 2 describes the steps for robot localization stage. Figure 3.11a shows the recognized landmarks obtained from the previous view (purple lines) and the current view (green lines) in the current view's frame of reference. Based on the above heuristics, the longest landmark is chosen for localization. The small red circle

Algorithm 2 Robot localization, localize $(\cdot)$

Input: (i) Recognized landmark list $RL_n = \{S_k^n\}$;

(ii) Perceptual map $PM = \{S_1, S_2, \cdots, S_i\}$;

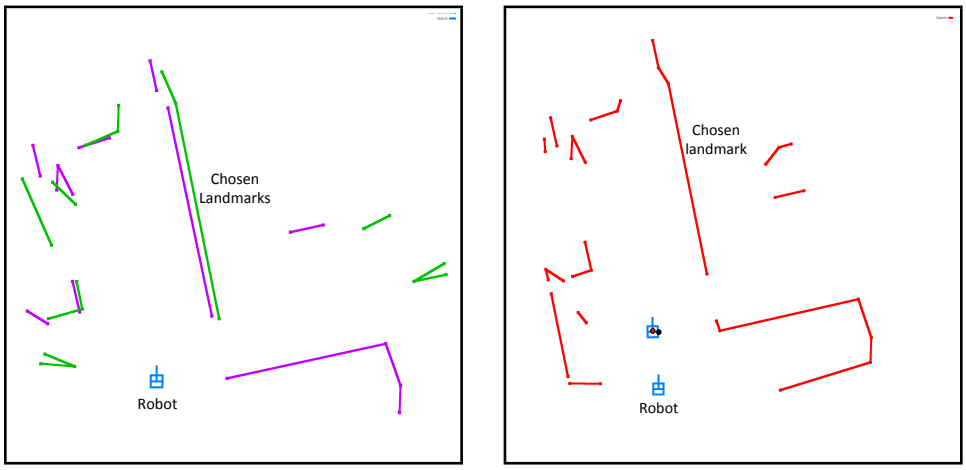
Output: A surface, $RP_n = \{P_1(x, y), P_2(x, y)\}$, where P_1 represents the robot's position and direction, P_1 to P_2 represents the robot's facing in the map.

Begin

1: Extract the best current view landmark (based on goodness value) from S_k^n and corresponding landmark from map, PM.
2: Set the robot's position in current view, $RP_{cv} \leftarrow S\{P_1(0, 0), P_2(0, 50 \text{ cm})\}$.
3: Calculate $dp_1, dp_2, \theta_1, \theta_2$ related to the current view landmark (for details see figure 3.10).
4: Calculate P_1 and P_2 by using equations 3.6 and 3.7.
5: Save the robot's position in the map, $RP_n \leftarrow S\{P_1, P_2\}$.

End

in figure 3.11b indicates Albot$_1$'s position calculated by the localization process using the chosen landmark, whereas the black circle indicates Albot$_1$'s position based on odometric information.



(a) (b)

Figure 3.11: (a) Recognized landmarks in the current view coordinate. (b) Calculated robot position in the map using equation 3.6 and 3.7.

3.1.7 Updating the Perceptual Map

Once Albot$_1$'s position in the map is known, it could transform the positions of the surfaces in its current view to their respective positions in the perceptual map. However, it computes the position of all new surfaces in the map using recognized landmark as a reference (i.e. same as locating its position in the map). In the next step, all surfaces in the map that lie within the boundary of the current view are removed. This is because the latest view is considered to provide a better description of these surfaces. However, there is an exception. Surfaces with parts lying outside of the boundary are not removed; rather they are extended to maintain a description of the entire surface perceived across multiple views. Formally, algorithm 3 describes the updating stage.

Figure 3.12 shows the inputs, output at intermediate stage, and the final map of the updating process. Figure 3.12a shows the input perceptual map (in this case previous view) which is about to be updated according to the current view information as shown in figure 3.12b. Figure 3.12c shows the current view boundary (dotted purple line) after transforming the current view onto the map. Old overlapped surfaces (green lines) are going to be removed from the map as shown in figure 3.12d. As surface S_m is viewed partially from the current location, it is going to be updated according to the first surface of the current view, S_1 (see figure 3.13 for another example). The perceptual map after removing surfaces corresponding with surfaces in the current view is shown in figure 3.12e. Figure 3.12f shows the updated map after adding a new view to it.

Finally, Algorithm 4 at the end of this section describes the formal algorithm implemented on Albot$_1$ to compute its perceptual map.

Algorithm 3 Updating perceptual map, update PM $(\cdot)$

Input: (i) Perceptual map, $PM = \{S_1, S_2, \cdots, S_i\}$;

(ii) A view, $V_n = \{S_1, S_2, ..., S_j\}$; and

(iii) Recognized landmark list, $RL_n = \{S_k^n\}$

Output: Updated perceptual map, $PM = \{S_1, S_2, \cdots, S_l\}$.

Begin

Extract the best current view landmark (based on goodness value) from $\{S_k^n\}$ and corresponding landmark from map, PM.
for all S_j **do**
    Calculate $dp_1, dp_2, \theta_1, \theta_2$ related to the current view landmark (for details see figure 3.10).
    Calculate P_1 and P_2 using equation 6 and 7.
    Save the transformed surface, $V_n' \leftarrow S\{P_1, P_2\}$.
end for
Compute a polygon by joining the edge points of V_n', polygon $\leftarrow V_n'\{S_1, S_2, \cdots, S_j\}$.
for all S_i **do**
    if P_1 and P_2 of S_i is inside the polygon or within 40 cm of the boundary **then**
        Delete S_i from PM.
    else if one edge point of S_i is inside the polygon and the other is outside the polygon
        Update S_i according to corresponding S_j' and delete S_j' from V_n'.
    end if
end for
Add all remaining surfaces from V_n' to PM, $PM \leftarrow$ add (V_n').

End

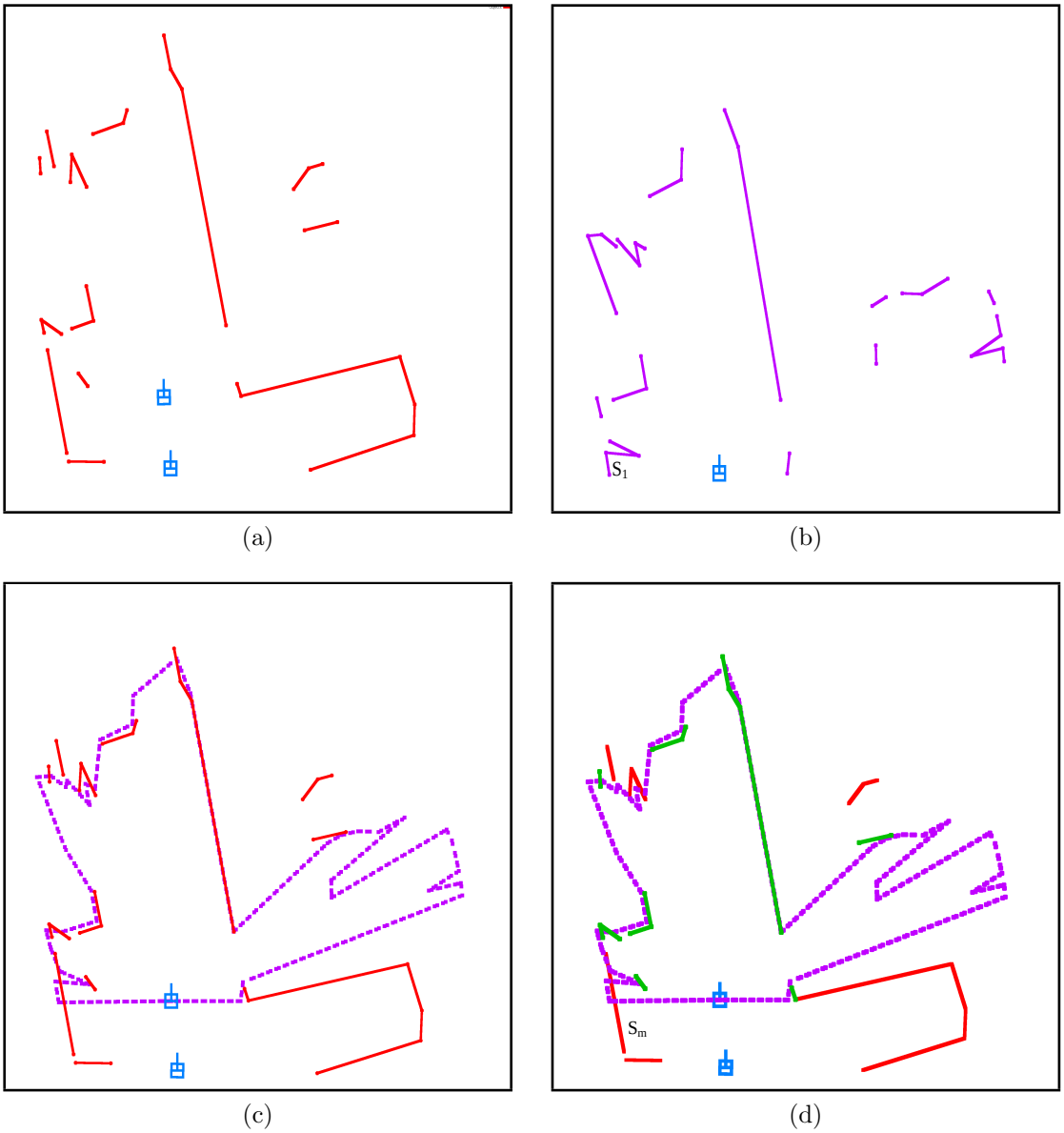


Figure 3.12: (a) The perceptual map (PM) prior to being updated with and (b) the current view. (c) Overlaying the current view boundary (dotted line) onto PM and (d) surfaces (green lines) inside the current view will be removed except S_m.

(e) (f)

Figure 3.12: (e) PM after removing those surfaces and (f) PM after adding surfaces from the current view and updating surface, S_u.

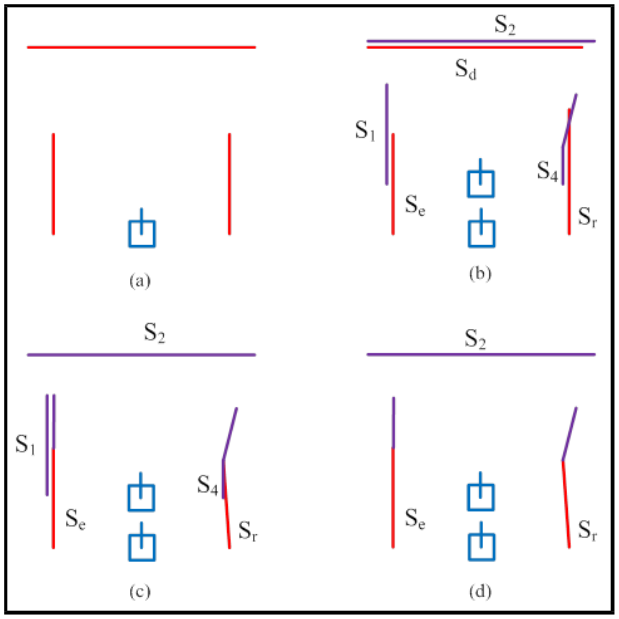


Figure 3.13: (a) The perceptual map (PM) is going to be updated, (b) new surfaces (purple line) are transformed on to PM, (c) S_e is extended as surface S_1, S_d is removed, and S_r is reduced as S_4, and (d) updated map.

Algorithm 4 Albot$_1$'s perceptual mapping

Input: A data file containing a set of laser scans (2D coordinate values of obstacles relative to robot's position from where that particular laser scan has been captured) and odometer information representing all the robot's positions.

Output: perceptual map, PM.

Begin

for all steps n **do**
  Capture laser scan: $LS_n \leftarrow (P_1, P_2, P_3, \cdots, P_i)$.
  Extract lines and construct a view: $V_n \leftarrow$ constructView (LS_n) (Any line extraction algorithm can be applied).
  Extract landmarks: $L_n \leftarrow$ findLandmark (V_n) (A surface is consider as landmark if it has at least one occluding point and is 40cm long).
  if $n = 1$ **then**
    Initialize perceptual map: $PM \leftarrow V_n$.
    Save this updating step/limiting point: $u_{step} \leftarrow n$.
    Initialize recognized landmark list: $RL_n \leftarrow L_n$.
  else
    Save robot's displacement (translation and rotation) relative to last step: $RD_{TR} \leftarrow$ {distance, angle}.
    Recognize the previous landmarks in current view: $RL_n \leftarrow$ recognizeLandmark(RL_{n-1}, L_n, RD_{TR}).
    if $(RL_n).size < 3$ **then** (update)
      if $(n - u_{step} = 1)$ **then** (update using the current view)
        Triangulate the robot's position in the map: $CRP_n \leftarrow$ localize (PM, RL_n).
        Update the perceptual map: $PM \leftarrow$ updatePM (PM, V_n, RL_n).
        Save this updating step/limiting point: $u_{step} \leftarrow n$.
        Reset recognize landmark list: $RL_n \leftarrow L_n$.
      else (update using the last view)
        Triangulate the robot's position in the map: $CRP_{n-1} \leftarrow$ localize (PM, RL_{n-1}).
        Update the perceptual map: $PM \leftarrow$ updatePM (PM, V_{n-1}, RL_{n-1}).
        Save this updating step/limiting point: $u_{step} \leftarrow n - 1$.
        Reset recognize landmark list: $RL_{n-1} \leftarrow L_{n-1}$.
        Go back to step 11
      end if
    end if
  end if
end for

End

3.2 Experiments and Results

This section describes six experiments designed to evaluate and test the robustness of the algorithm in an indoor office-like environment.

3.2.1 Experiment 1: The Map Computed

The first experiment is designed to evaluate how well Albot_1 computes its perceptual map. Note that given that the map computed is imprecise and inexact, "loop closing" is not a suitable test of how good/accurate the map is. Although "loop closing" is a test that is popular in robot mapping, instead, and given that it is well known that robot mapping without error correction will result in a seriously distorted map, we claim that Albot_1 is successful if its map maintains an overall shape of the environment traversed. While such a measure is subjective, it suffices for now given that the nature of such a map is little understood.

Figure 3.14 and 3.15 portrait six different routes Albot_1 traversed through the environment and generated maps. The results show that all the maps maintain an overall shape of the environment traversed. However, in the first test as shown in figure 3.14 (top left), Albot_1 revealed an interesting situation when it travels down the corridor indicated by C. It is a long narrow corridor with two side surfaces only. It appears to be featureless to Albot_1. Albot_1 could not recognize any landmark to localize itself and was force to depend on path integration process (localizing and updating map using odometer data for every step). Consequently, the representation of that corridor is more distorted than other parts of the map.

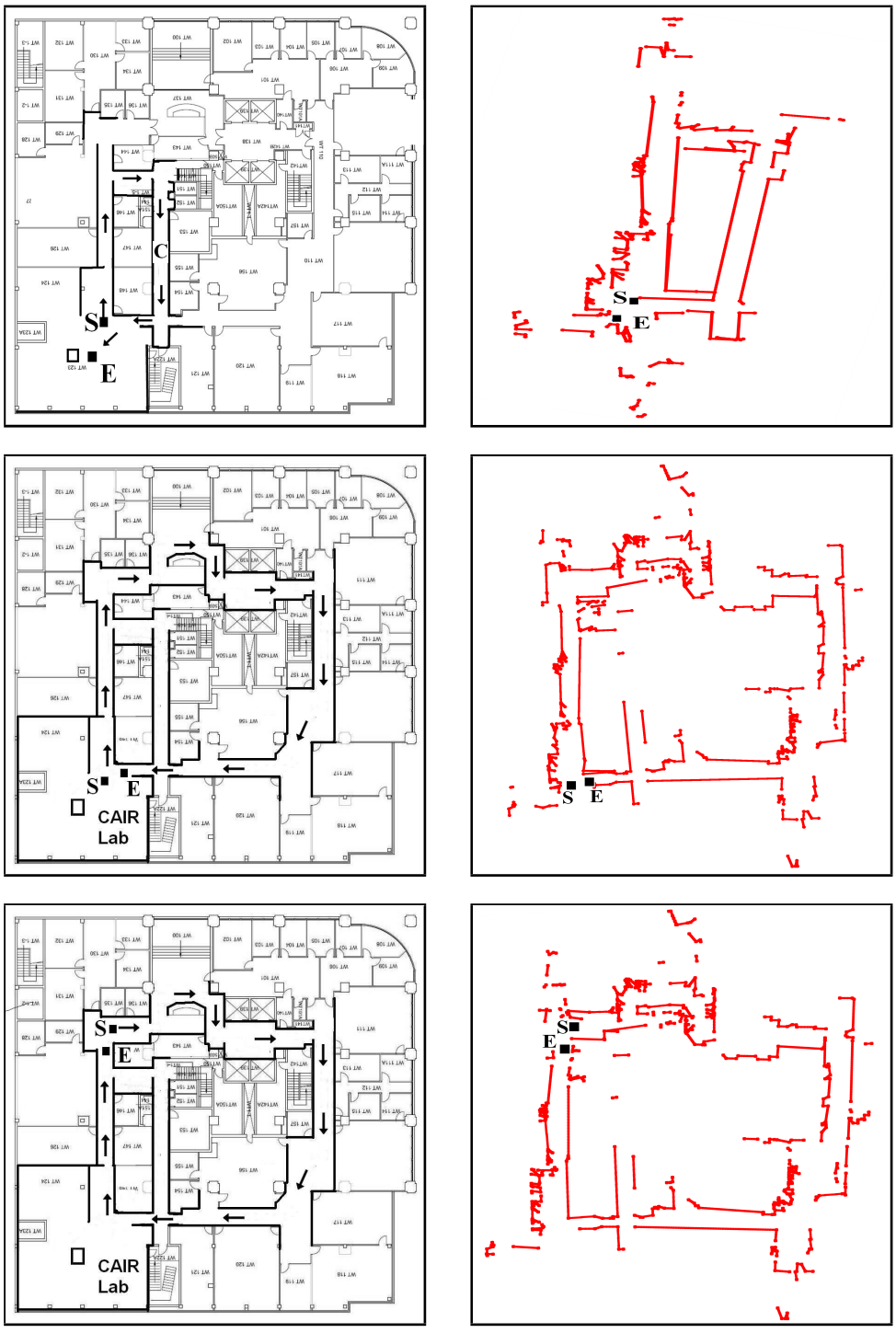


Figure 3.14: Maps generated by Albot$_1$ for three tests. The route is indicated by arrows (left image) and corresponding map computed (right image).

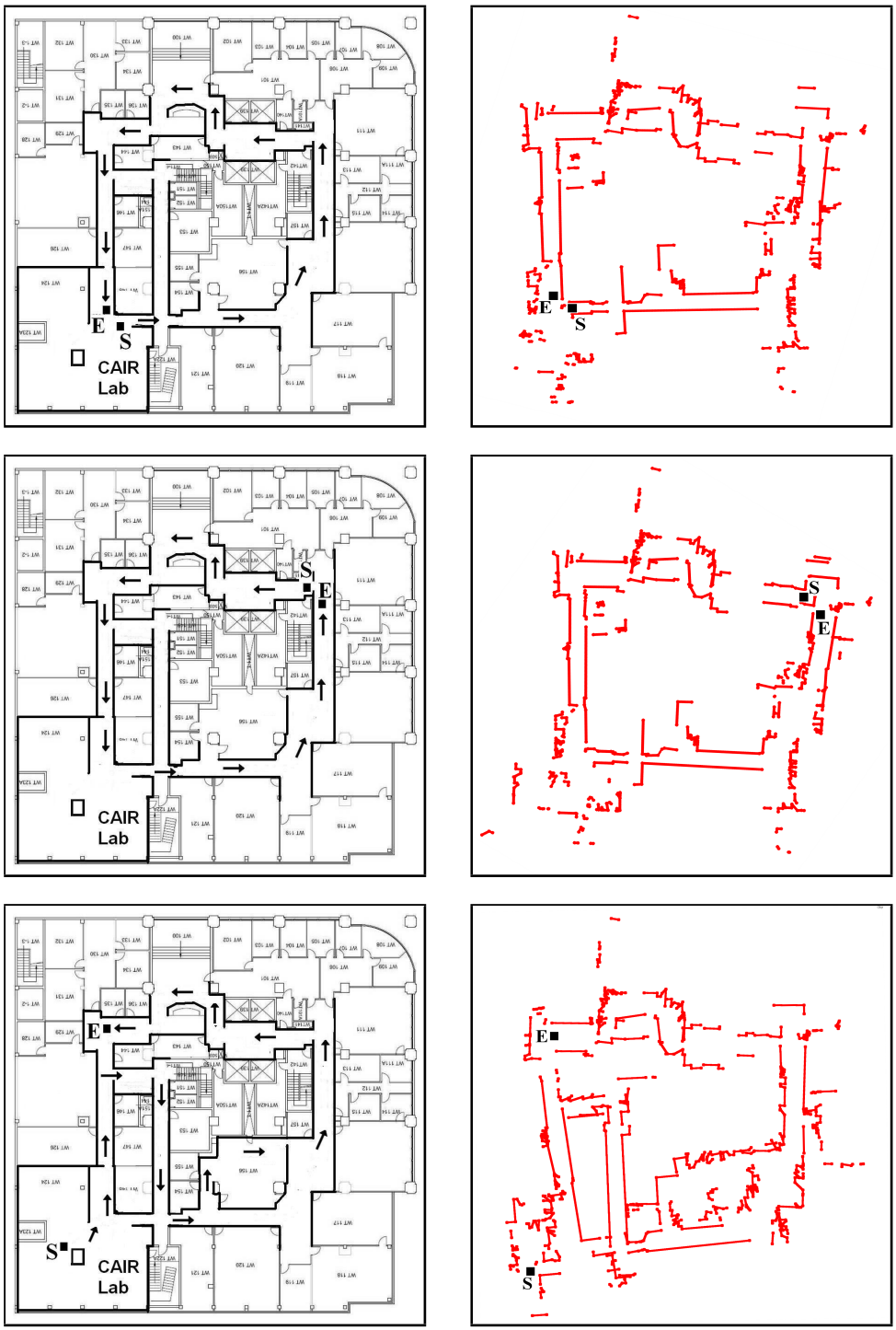


Figure 3.15: Maps generated by Albot$_1$ for three tests. The route is indicated by arrows (left image) and corresponding map computed (right image).

Figure 3.16 shows what happens when Albot_1 re-visited a previously visited part of the environment. Given that there is no matching of old and new information, that part of the environment is simply replaced with what is currently in view. From a

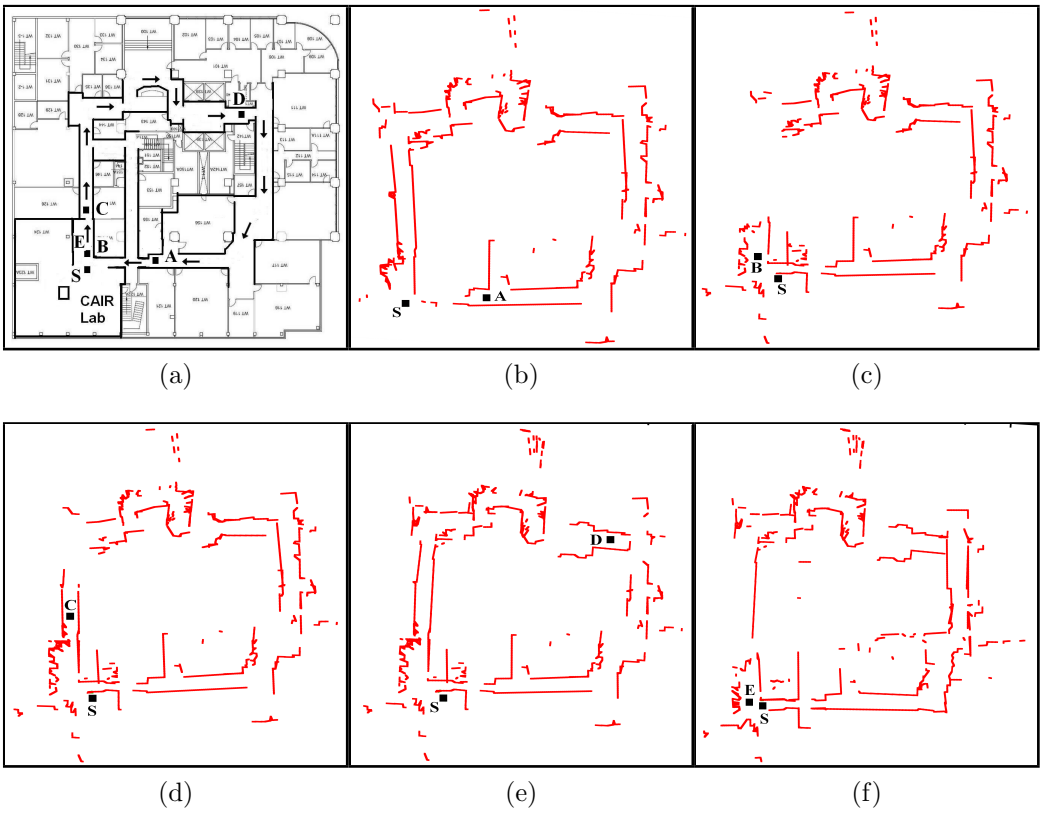
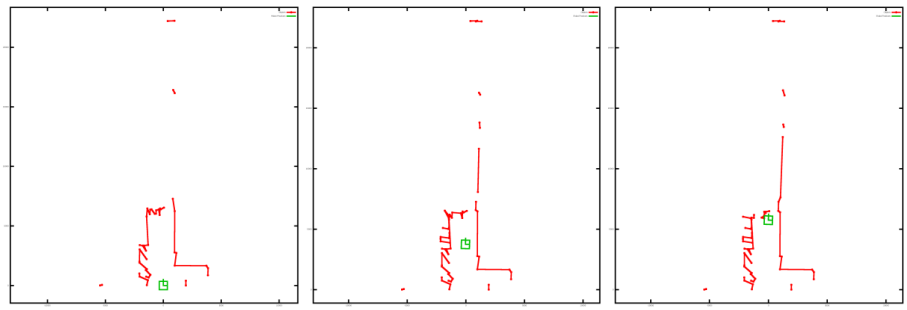


Figure 3.16: (a) A test environment. Albot_1 started traveling from point S and followed the arrow directed path. (b) Map computed when Albot_1 is at point A just before revisiting, (c) at point B upon revisiting, deleting old corridor information, (d) at point C building the corridor again, (e) at point D, and (f) returned near starting position.

cognitive agent's perspective, it is important to use the current view of the environment as this is the most up-to-date description of this part of the environment. What is held in one's memory might become obsolete. Since the goal here is not to build an integrated map, one does not integrate what one sees with what one remembers.

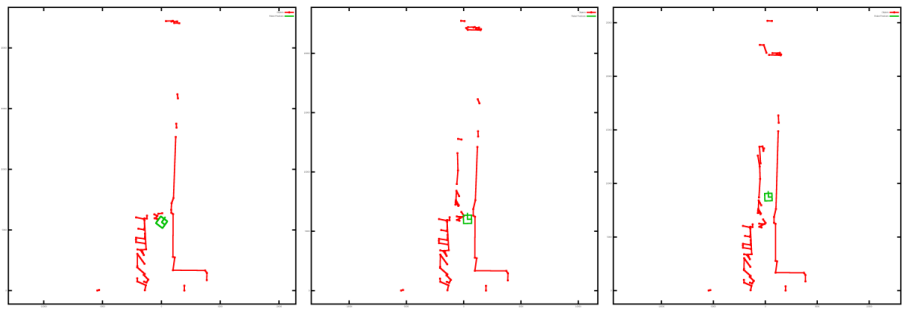
In this sense, $Albot_1$'s map is considered to be expanding ad infinitum and what is important is that that at all times the overall shape of the map is maintained.

Figure 3.17 provides a trace of $Albot_1$'s mapping process. In particular, it shows how often its map is updated and the effect of removing surfaces in it prior to updating it with a view. For example, when moving down the corridor (figure 3.17f-l), the map is updated seven times. This is because the surfaces chosen for tracking are often close to $Albot_1$ (in this case, there are few landmark surfaces) and as a result, the map of the corridor produced is detailed. In another situation, when going through a more open space (figure 3.17 v-x), the map is updated three times for ten views and as a result the map of this space is less detailed. For example, in figure 3.17w, there is a large incomplete area marked by dashed rectangle even though $Albot_1$ saw this part of the environment.

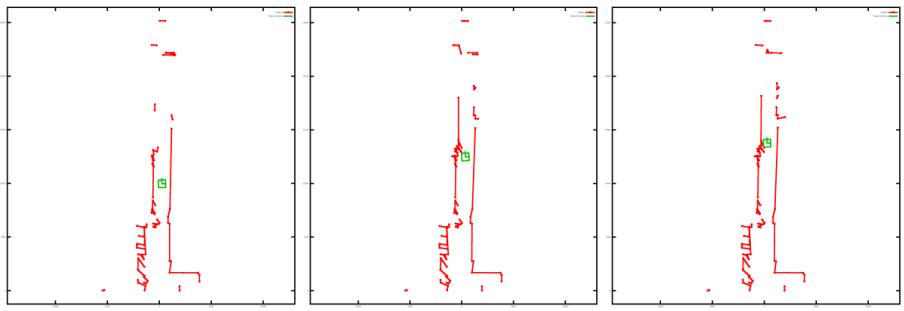


(a) start (b) 3 (c) 2

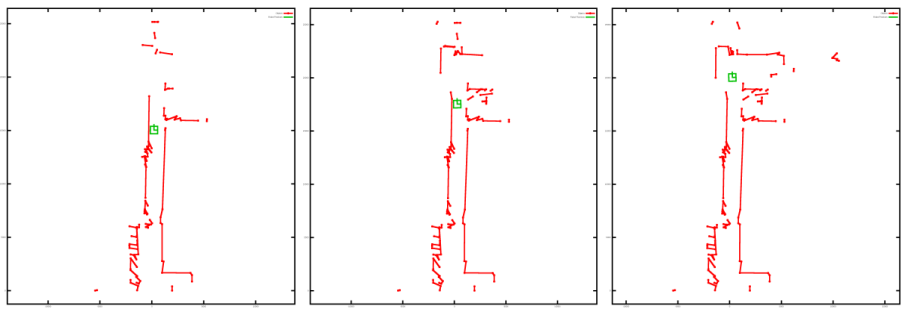
Figure 3.17: (Part1 - three updates) How $Albot_1$'s perceptual map changes - each number indicates the number of move/turn instructions executed before an update is required. Each move is, on average, about 3m.



(d) 1 (e) 2 (f) 2

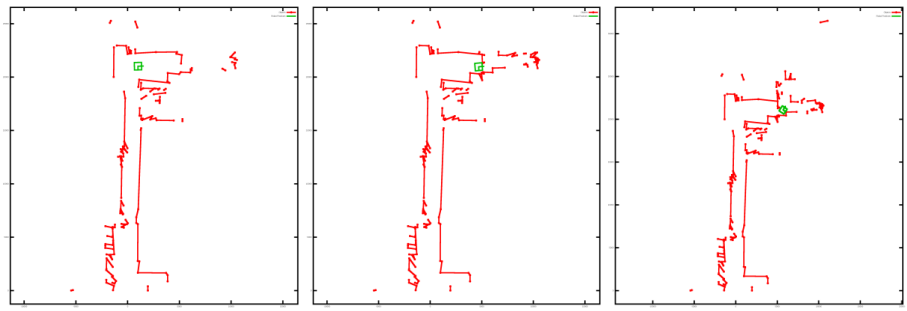


(g) 1 (h) 2 (i) 1

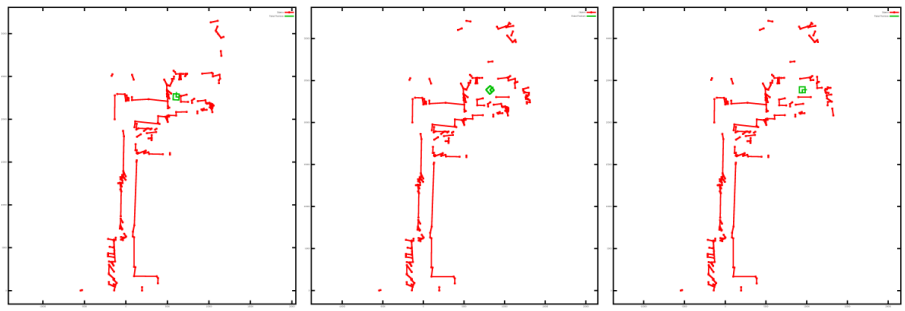


(j) 1 (k) 2 (l) 2

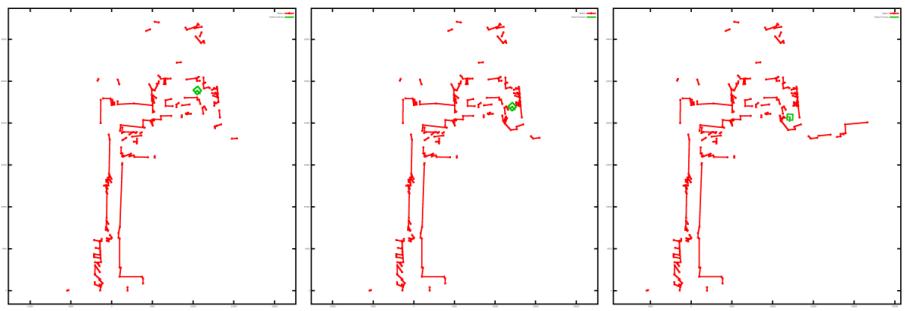
Figure 3.17: (Part2 - nine updates) How Albot$_1$'s perceptual map changes - each number indicates the number of move/turn instructions executed before an update is required. Each move is, on average, about 3m.



(m) 3 (n) 3 (o) 2

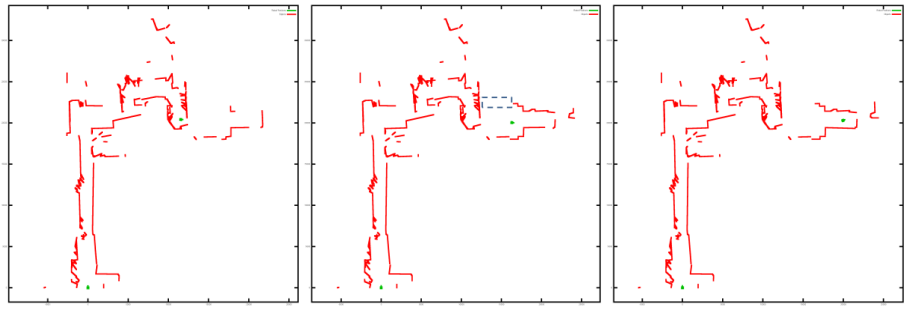


(p) 3 (q) 3 (r) 2



(s) 2 (t) 2 (u) 2

Figure 3.17: (Part3 - nine updates) How Albot$_1$'s perceptual map changes - each number indicates the number of move/turn instructions executed before an update is required. Each move is, on average, about 3m.



(v) 4 (w) 4 (x) 4

Figure 3.17: (Part4 - three updates) How Albot_1's perceptual map changes - each number indicates the number of move/turn instructions executed before an update is required. Each move is, on average, about 3m.

3.2.2 Experiment 2: Orientation

Even though the map computed is imprecise and inexact, how good/accurate is it? In this experiment, we provide one means of measuring the accuracy of Albot_1's map: how well does Albot_1 orient itself in the environment? During two journeys through the environment, each using a different route, Albot_1 calculated the distance and orientation of its starting position from two different stop positions (see figure 3.19a-b and 3.19c-d respectively). Corresponding measurements were then made using the floor plan (figure 3.18) of the physical environment and Albot_1's positions in it. The latter is done by first using Albot_1's perceptual map to compute its relative position from a surface that is visible to it in its stop position. Then, the same surface is identified in the floor plan and Albot_1's position in the floor plan is then calculated from its position relative to that surface. We calculate Albot_1's starting position in the floor plan using the same approach. The results are shown in Table 3.1. The

orientation angle is a measure of difference between Albot$_1$'s current facing position and the starting position; its sign indicates if Albot$_1$ must turn left (+) or right (-).

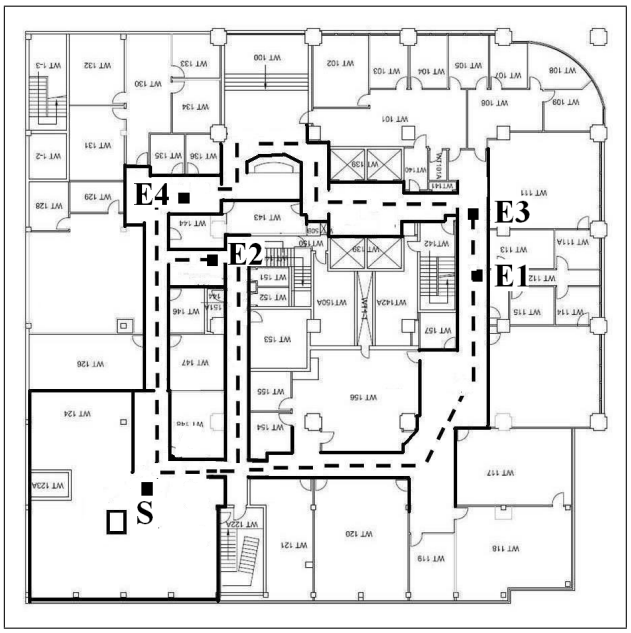


Figure 3.18: A test environment. Albot$_1$ started traveling from S four times in two directions and stopped at E1, E2, E3, and E4 to measure distance and angle of the starting location.

	Albot$_1$'s map	Physical Map
3.17a	-75.50/25.8m	-70.10/27.3m
3.17b	83.00/19.6m	77.00/16.95m
3.17c	114.00/32.1m	117.50/35.3m
3.17d	82.50/21.6m	76.00/23.0m

Table 3.1: Angle and distance of the starting point measured from four different locations.

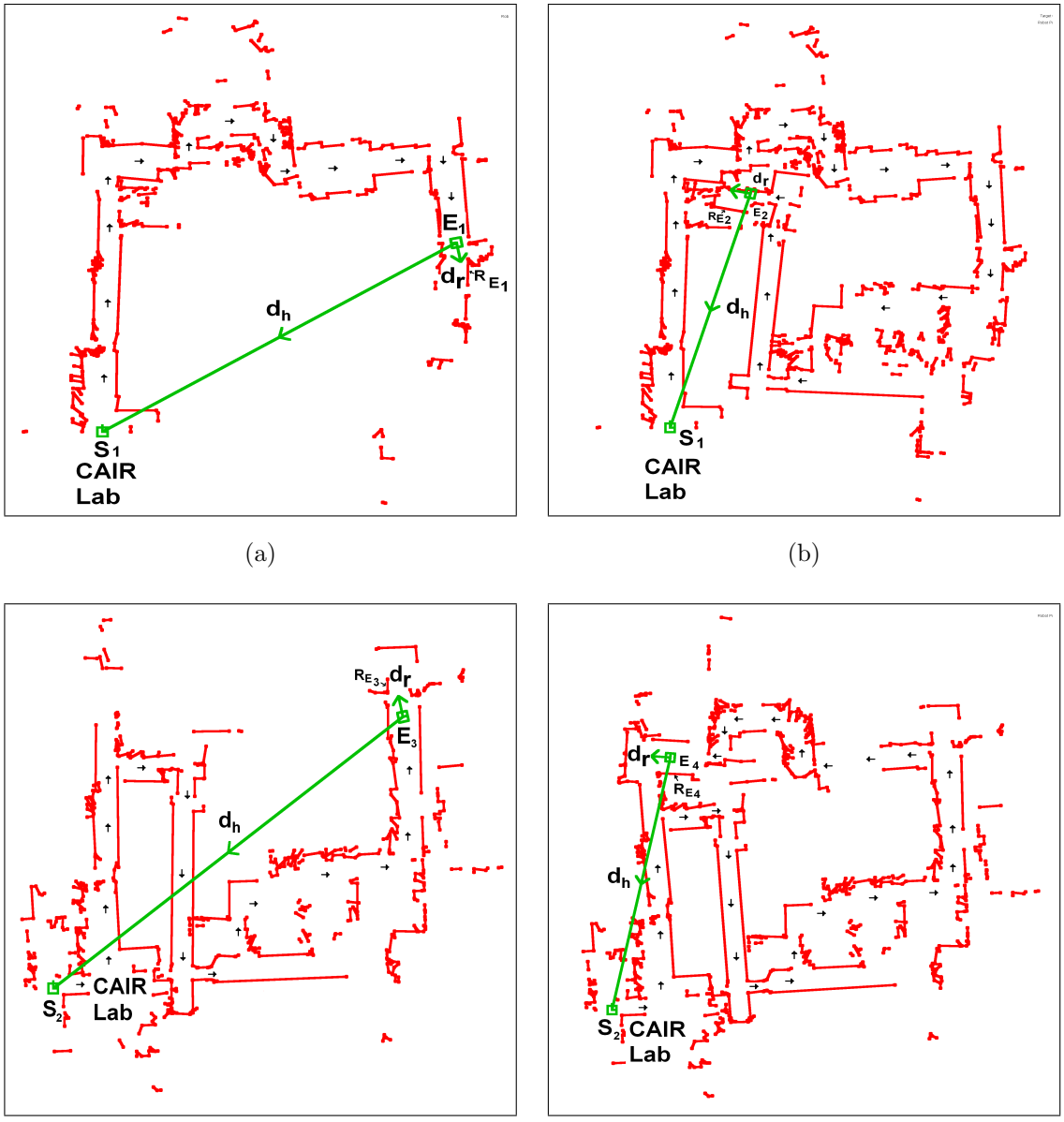


Figure 3.19: Orientation Experiments - Albot$_1$ is pointing towards the starting location.

Given that the difference in angle and distance are no more than +/-7^0 and +/-3m

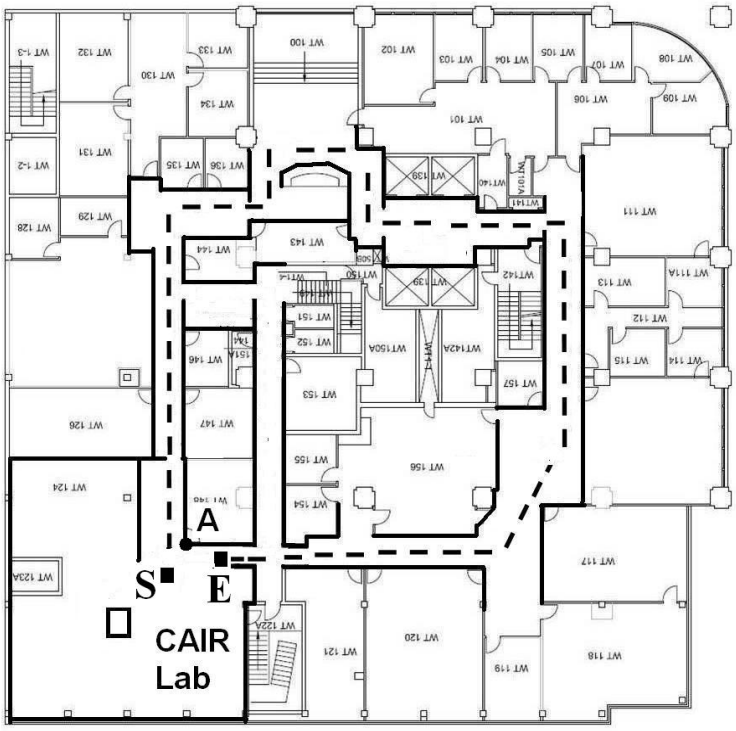
respectively, we conclude that Albot_1 is able to use its map to orient itself well in its environment.

3.2.3 Experiment 3: Error Tolerance

We know that the absolute position of the robot provided by its odometer is not error free. During translational movement, any slight change of conditions on the ground surface will cause the robot to change course slightly even though the robot believes that it is still moving in a straight-line. In addition, another source of error comes from its caster wheel. During rotation, the caster wheel provides resistance to movements. As a result, the robot rotates less than it intends to. Consequently, for both translational and rotational errors, the robot gets a distorted representation of its environment based on odometer information alone. In this experiment, we investigate how well Albot_1 tolerates erroneous odometer information when computing its map.

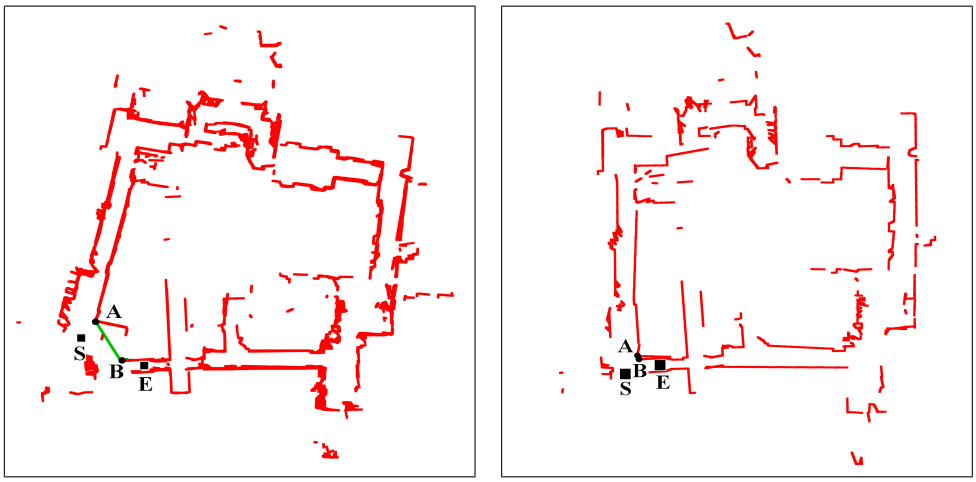
During this experiment, Albot_1 traveled approximately 100 meters starting from point S to point E following the route indicated by arrows as shown in figure 3.20 and collected 106 views. Figure 3.21a shows the map computed using simple mathematical transformations (transforming all individual views in a single co-ordinate frame using odometric information). The green line indicates resultant distance error (approximately 4.6 meters) in the map as point A and B represent the same corner location (point A in figure 3.20) in the real environment. Figure 3.21b shows the map computed by Albot_1's algorithm, where the final distance error is 0.4 meter. In the next test, we added 0.1 meter and 1 degree to translation and rotation data of two consecutive robot locations to induce extra error. Figure 3.22a shows the map computed by

the simple mathematical transformation, where the green line indicates resultant distance error (approximately 22.3 meters). Figure 3.22b shows that Albot$_1$'s algorithm still can compute a good map (resultant distance error approximately 1.6 meters).



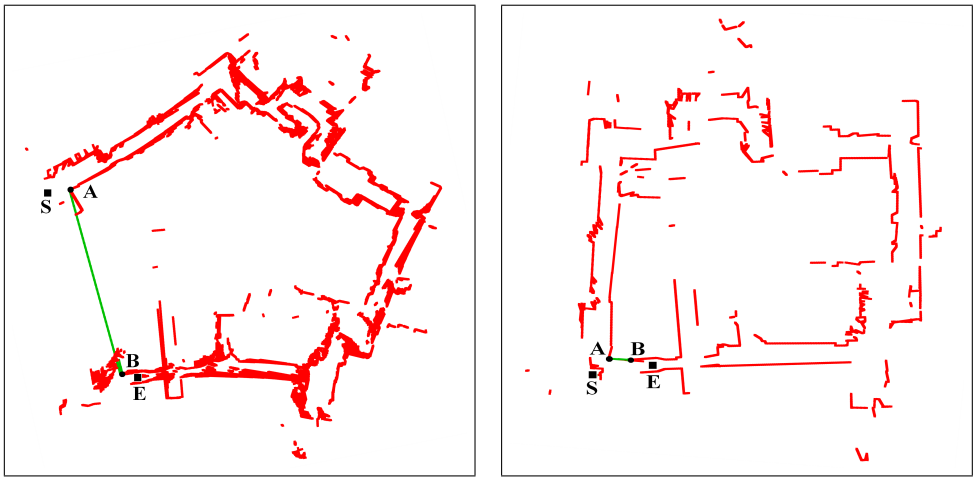
(a)

Figure 3.20: The test environment. The dash line indicates the path traveled by Albot$_1$.



(a) (b)

Figure 3.21: Map computed using (a) a simple mathematical transformation and (b) Albot$_1$'s algorithm. During this experiment, the robot traveled approximately 100 meters from point S to point E following a square-like route. Point A and B are same location in the real world. In figure a, the green line represents perceptual error (approximately 4.6 meters) in input data.



(a) (b)

Figure 3.22: Map computed using (a) a simple mathematical transformation and (b) Albot$_1$'s algorithm. In figure a, the green line represents perceptual error (approximately 4.6 meters) in input data.

3.2.4 Experiment 4: Continuous Movement

Until now, all the test runs were done with Albot$_1$ moving in a start-stop manner. In this experiment, Albot$_1$'s movements were guided by moving the arrow keys of the attached laptop. A data log program is used to save laser scans and odometric positions for every 10cm translation or 1 degree rotation. Figure 3.23 shows the path taken by Albot$_1$: $S \rightarrow E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow E_4 \rightarrow E_5 \rightarrow E_6 \rightarrow E$ where E and S are the same locations. A dataset is obtained consisting of 1330 views and their corresponding odometric robot positions. Each consecutive view is either 10cm or 1 degree apart.

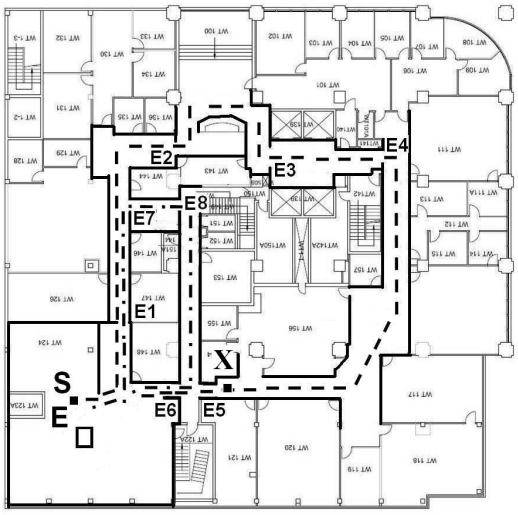
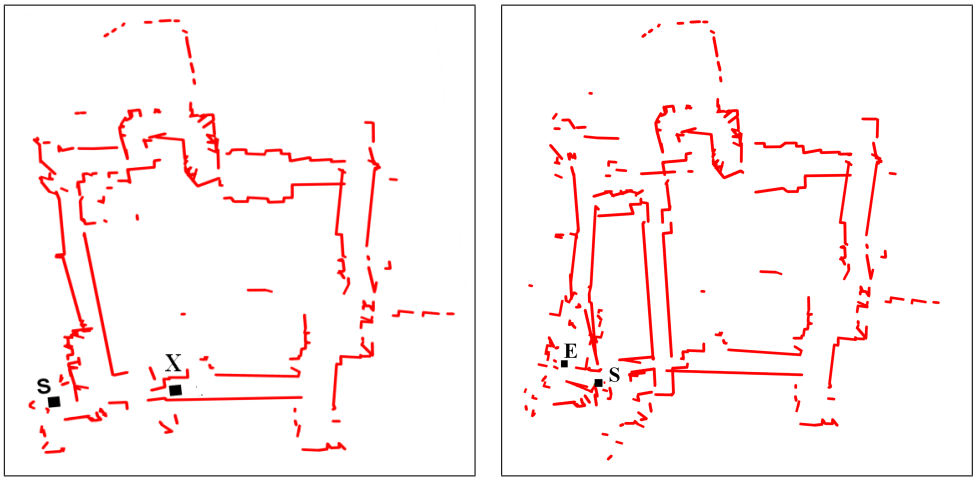


Figure 3.23: The test environment.The dash line indicates the path (approximately 136 meters measured by odometer) travelled by the robot.

The results of this experiment are shown in figures 3.24. Figure 3.24a shows the map computed using our algorithm, when the robot stops at point X, just before crossing exit E5. Up to this point, the robot traversed approximately 95 meters (measured by its odometer) and obtained 810 views. Figure 3.24b shows the final

map produced, when the robot returns back to its starting location S after revisiting some parts of the environment. At this point, the robot has obtained 1330 views and updated the map only 470 times, yet it preserves the overall shape of the environment well. It should be noted that during perceptual mapping, our robot does not detect loops. As described in the theory, when the robot returns to a familiar part of the environment, it will simply overwrite surfaces corresponding with what is in the map with the latest information perceived. However, how information is removed from the map prior to updating will have an effect on what is left in one's memory. If too much is removed from the map, one could be removing useful information about the nearby environment that one has previously seen, but if too little is removed, one could have "ghost" surfaces remaining in the map. In the current implementation, we remove a space slightly larger than the incoming view and this enables us to generate a clean map.



(a) (b)

Figure 3.24: The map produced using Albot$_1$'s algorithm: (a) before revisiting and (b) after completing journey.

3.2.5 Experiment 5: Comparing with SLAM

Given that Albot_1 now computes its map while exploring its environment continuously, we could collect its dataset and run other algorithms over it. Conversely, we could test Albot_1s mapping algorithm using other datasets. In this experiment, we compare Albot_1s algorithm with some well-known SLAM algorithms downloaded from the Internet (Stachniss et al., 2007; Montemerlo et al., 2002).

Our first test is to run our dataset with 4 well-known SLAM algorithms, namely Carmen mapping which uses Monte Carlo Localization based on particle filters and scan matching techniques (Thrun, Fox, Burgard, & Dellaert, 2000), DP-SLAM which uses distributed particle filter (Eliazar & Parr, 2003), gridSLAM which uses Rao-blackwellized particle filter and scan matching (Hahnel et al., 2003), and GMapping which uses Rao-Blackwellized particle filters (Grisetti et al., 2007). We expect that these SLAM algorithms will produce very accurate maps of the environment. Our results show that this is the case (see figure 3.25).

Using the same dataset, without the need to update using every consecutive view, our robot could move and track landmarks in view, and prior to their disappearance, update its map. However, we could not demonstrate such tracking with a laser as our primary sensor. Instead, in this experiment, we test our robot computing such a map for route $S \rightarrow E_1 \rightarrow E_2 \rightarrow E_3 \rightarrow E_4 \rightarrow E_5 \rightarrow E_6 \rightarrow E$. Similar to experiment 4, the robot traversed approximately 136 meters (measured by its odometer). During this journey, another dataset is obtained consisting of 217 views and corresponding odometric robot positions, where each consecutive view is either 1m or 10 degrees apart.

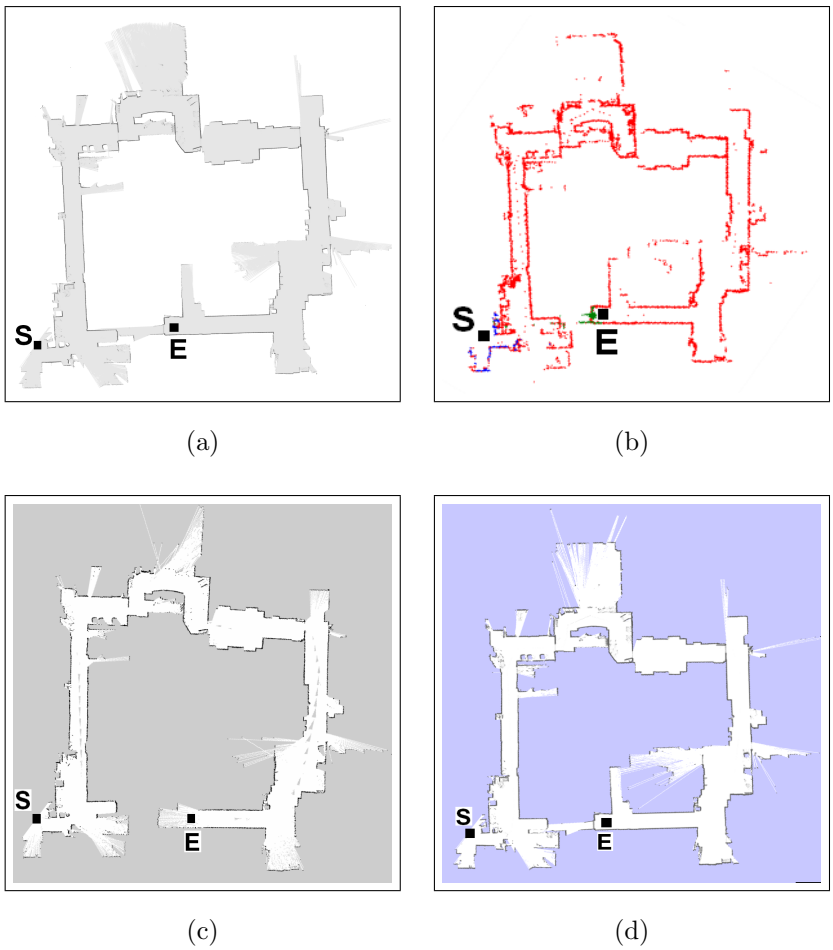
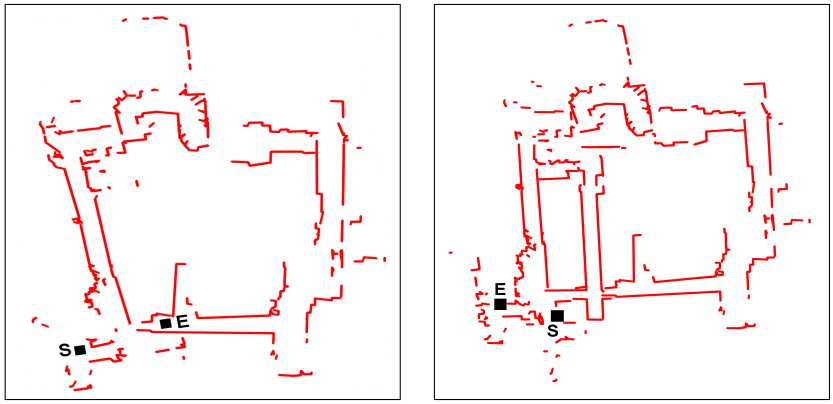


Figure 3.25: Maps computed using (a) DP-SLAM, (b) Carmen mapping, (c) GMapping, and (d) gridSLAM.

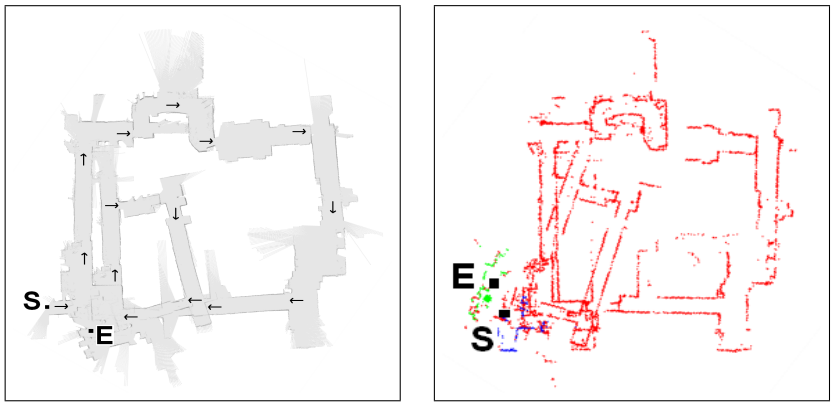
The results of this experiment are shown in figure 3.26, 3.27 and 3.28. figure 3.26a shows the map produced just before our robot enters into the loop. At this point, the robot obtained 128 views and updated its map 76 times. The map is similar to that produced during experiment 4 (figure 3.24a) even though it has many fewer views of the environment. Figure 3.26b shows the final map produced by updating 133 times for all 217 views. Interestingly, the robot gets a better orientation of the corridor

being re-visited and the overall shape of the environment is better than before. None of the SLAM algorithms, however, produce a correct map. As shown in figure 3.27 and 3.28, their maps are completely distorted. Again, it is not too surprising since all these algorithms depend on good data association and handling of errors.



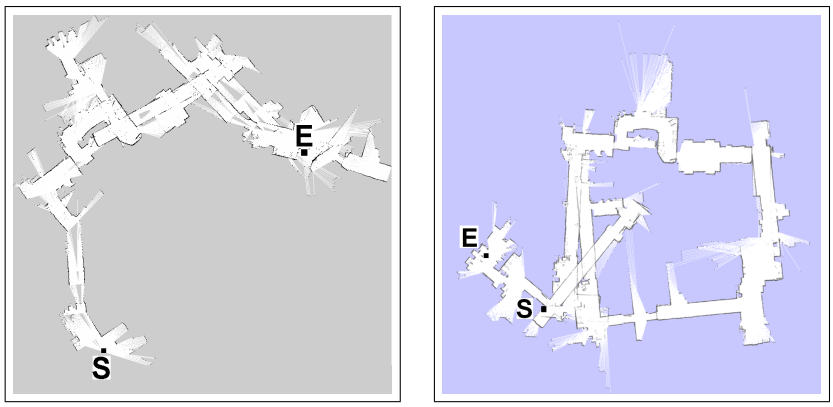
(a) (b)

Figure 3.26: Maps computed using our algorithm (a) before entering a loop and (b) after completing the journey.



(a) (b)

Figure 3.27: Maps computed using (a) DP-SLAM and (b) Carmen mapping.



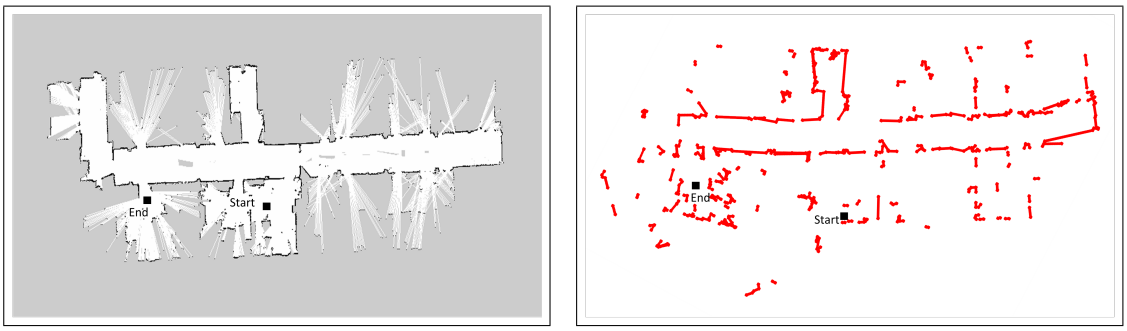
(a) (b)

Figure 3.28: Maps computed using (c) GMapping and (d) gridSLAM.

3.2.6 Experiment 6: Mapping from Online Dataset

So far, we demonstrated Albot$_1$'s ability to compute a map using a dataset it collected. In addition, we compared the results of Albot$_1$'s algorithm with SLAM algorithms using the same dataset. Following this, we designed an experiment to test whether Albot$_1$'s algorithm can compute a map from a dataset which is available from the SLAM community. We used a dataset which is both recent and consists of laser reading and odometric information similar to Albot$_1$'s input.

The results of this experiment are shown in figure 3.29. Figure 3.29a shows the map computed using GMapping algorithm (Grisetti et al., 2007). During this data collection, the robot started travelling from a location indicated by "Start", went up to the left end of a large corridor, turned and continued up to the right end the corridor, turned and moved to the location indicated by "End". Figure 3.29b shows that Albot$_1$'s algorithm can compute a map in reasonable shape.



(a) (b)

Figure 3.29: Map computed using (a) GMapping alborithm (Grisetti et al., 2007) and (b) Albot$_1$'s algorithm.

3.3 Discussion

This chapter shows how Albot$_1$ computes a transient, imprecise, and incomplete perceptual map of an office-like environment using laser and odometric information. As implemented, algorithms used do not employ any error correction method. In addition, since the map is not updated continuously, there is no merging of information during updates, i.e. between information in the map that is found to co-locate with information from the incoming view. Previous information could be entered into the map during an earlier visit or from a different vantage point. Such information is removed from the map before information from the incoming view is added, and thus, misalignment of the map is not detected. This makes updating the map straightforward and allows for the map to be expanded ad infinitum. The resulting perceptual map is a transient and incomplete map and represents a trace of one's movement rather than a map of one's total experience.

To demonstrate the robustness and effectiveness of the implemented algorithm, six experiments have been conducted. Experiment 1 shows that the map is not updated

for every successive view; it is updated when the robot leaves a local environment and enters another local environment. As a result, the map computed is incomplete. Experiment 2 shows that even if there is no error correction method, Albot_1 maintains a map of reasonable shape and can point to the starting position quite well. Experiment 3 shows that even if we induce extra error in odometric data, the algorithm still can compute a map of reasonable shape. Experiment 4 and 5 shows that when two consecutive views are 1 meter or 10 degree apart, Albot_1 performs better than traditional SLAM based algorithms in maintaining the shape of environment. In addition, experiment 6 shows that Albot_1 can compute a map of reasonable shape from external datasets which are being used for robotics research.

In summary, this chapter illustrates the overall performance of Albot_1's perceptual mapping algorithm. It also shows how different Albot_1's algorithm is compared to SLAM based algorithms. In particular, there is no recognition of places re-visited or loop closing at the perceptual level. Recognition of places is not performed, because the purpose of Albot_1's perceptual map is to orient with respect to a starting position and remember its immediate surroundings. We recognize that this information is necessary to create and maintain an enduring map. In the next chapter, we describe how Albot_1 computes an enduring representation from this perceptual map.

Chapter 4

Albot$_1$'s "Cognitive" Map

Albot$_1$'s perceptual map is a transient map; information in it will disappear over time. As such, and as cognitive researchers suggest (see Chapter 2), information in it needs to be transferred to a more enduring map. In this chapter[1], we extended our work on perceptual mapping and consider how a "cognitive" map might be formed from such a perceptual map. The word cognitive is in quote because Albot$_1$, with its limited sensing, has little ability to reason with what is out there and therefore the map computed is, at best, a very primitive cognitive map. However, one important aspect of cognitive mapping, as opposed to perceptual mapping, is the ability to perform abstraction and use the knowledge abstracted to help solve spatial tasks. For example, Kuipers' (2001) model of a cognitive map shows how, using abstractions, a complex hierarchy of place representations is built (see Beeson et al. (2010) for more recent work based on his model) and robotics researchers also compute a topological-metric map for path planning (Thrun, 1998; Poncela, Perez, Bandera, Urdiales, & Sandoval, 2002; Konolige, Marder-Eppstein, & Marthi, 2011).

In this chapter, we show how Albot$_1$ builds an abstract representation of places and

[1]This chapter has been published in Procedia - Social and Behavioral Sciences (Hossain & Yeap, 2013)

how such a representation becomes useful when navigating in a familiar environment. The notion of a place is more abstract than the notion of a local environment afforded in a view. While the latter is a bounded representation that is delivered directly at the perceptual level, the former can be real or imagined, and its boundary can be precise or fuzzy, and/or overlap (Golledge, 1999). A place is one's conceptual view of the environment grounded through one's experience in it. Its physical size should not be pre-determined using some fixed strategy. This is in contrast with the approach taken by robotic researchers who partition the environment into "areas of convenient sizes where [their] techniques can be applied efficiently to produce consistent local maps" (Blanco, Fernandez-Madrigal, & Gonzalez, 2008). Rather, Albot$_1$'s perceptual map holds its recent experience in the environment, and information in it is abstracted to be part of, but doesn't define, a place visited. A place is often learned without knowing its entirety.

To compute a place representation, Albot$_1$ needs to be able to compute a representation of places visited and to recognize those being re-visited. Two algorithms designed to fulfill this need are described in section 4.1. In section 4.2, three experiments were conducted to test Albot$_1$'s ability to compute its network of place representations. Section 4.3 discusses the significance of the results obtained.

4.1 Computing Albot$_1$'s Network of Places

This section presents two algorithms that enable Albot$_1$ to compute a network of place representation from its perceptual map. The first algorithm enables Albot$_1$ to compute a representation of places visited and the second algorithm enables Albot$_1$ to recognize places re-visited. The resulting topological map learned is a list of places

visited and each place consists of two pieces of information: an inexact metric map with its own frame of reference and an exit list containing information about exits used to move in and out of this place. Each exit has three values: its whereabouts in the current place, the place it leads to, and the exit of the place it leads to.

4.1.1 Remembering Places Visited

Humans and animals are observed to make use of a rich variety of information to define and recognize a place. Such information includes landmarks (Downs & Stea, 1973), local geometry (Cheng, 1986), functional properties and an array of unique/ interesting objects. Being a primitive "species", Albot_1 needs a mechanism to signal that it is in a place. Given that it explores an office-like environment with well-defined exits (i.e. doorway, a gap whose typical size is about 1 meter in Albot_1's environment), an algorithm computes a place representation whenever an exit is crossed. Algorithm 5 describes the steps for remembering places as shown on the next page.

Note that the last step creates a force update of the perceptual map and is done to re-enter the surfaces in view into the perceptual map so that they are marked as part of the current place.

This algorithm thus defines a place as the part of the environment one has experienced prior to crossing an exit and leaving the place. Initially, the algorithm will generate an incomplete place definition that also includes surfaces that are part of another place. It is incomplete because the robot seldom explores a place completely. Its initial definition will include surfaces of other places because these surfaces are perceived through the exits while the robot is still exploring the current place. As noted earlier, both these properties are part of a place definition.

Algorithm 5 Remembering Places Visited, rememberPlace $(\cdot)$

Input: (i) Perceptual map, $PM = \{S_1, S_2, \cdots, S_i\}$, where surfaces are marked as belonging to P_c when it is updated with new surfaces.

(ii) Crossed exit, $E_x = \{P_1(x, y), P_2(x, y)\}$, where the exit is.

(iii) Current place, $P_c = (\{\}, \{E_e\})$, where, $\{\}$ is an empty list of surfaces, and E_e is the exit through which Albot_1 came in this current place.

Output: Current place, $P_c = (\{S_1, S_2, \cdots, S_j\}, \{E_e, E_x\})$, where, $\{S_j\}$ is the list of j number of surfaces, E_e is the exit through which Albot_1 came to this current place, and E_x is the exit Albot_1 just crossed.

Begin

1: **if** an exit is crossed **then**
2: **for** all S_i **do**
3: **if** S_i is marked as a current place object **then**
4: Add S_i to current place surface list, $P_c \leftarrow$ add (Si).
5: **end if**
6: **end for**
7: Add E_x to current place exit list, $P_c \leftarrow$ add (E_x).
8: **if** P_c is a new entry in the network of places **then**
9: Create a new node in the map for P_c.
10: **else** (update place)
11: Combine its description with the appropriate entry in the network of place representation.
12: Update the exit information between this place and the previous place so that it makes a correct reference to this place node;
13: Check if the exit just crossed is leading to a known place, say, P_v. If so, mark P_c as equivalent to P_v.
14: **end if**
15: Set current place to $P_c = (\{\}, \{\})$;
16: Update perceptual map with the current view and update exit list of current place to include the exit just crossed as entering exit: $P_c = (\{\}, \{E_e\}) \leftarrow$ add (E_x).
17: **end if**

End

4.1.2 Recognizing Places Re-visited

With limited sensing, Albot_1 uses its perceptual map to detect returning to a familiar place. To do so, whenever it finds surfaces in its perceptual map that co-locate with those from the incoming view, it checks to which places these surfaces belong before it deletes them from its perceptual map. However, due to the inexact nature of the perceptual map, encountering such surfaces does not necessarily mean that one is in a familiar place. To increase the likelihood, Albot_1 seeks further evidence to confirm that this is the case. It does so by comparing the structural description of the current place with the remembered place that it believes it is currently re-visiting. Hence, whenever Albot_1 updates its map with the incoming view and finds there are more than n co-locating surfaces in the map, it will attempt to confirm if it is in a familiar place. n is a function of the size of the place being re-visited; a smaller place requires a lesser number of surfaces to be sighted prior to confirming that one is re-visiting it. The algorithm for confirming a place being re-visited is defined as shown on the next page.

Note that in the implementation, it is considered surfaces with a length $> 40cm$ and two surfaces are considered the same if they are not separated by a gap $> 40cm$ and the angle between them is $< 5^0$. These threshold values are chosen arbitrarily.

4.2 Experiments and Results

Three experiments are conducted to test and evaluate Albot_1's ability to abstract places from the perceptual map to form the network of places.

Algorithm 6 Recognizing Places Re-visited, recognizePlace $(\cdot)$

Input: (i) Surfaces, $P_{cs} = \{S_1, S_2, \cdots, S_j\}$, surfaces which belong to current place;

(ii) Surfaces, $P_{vs} = \{S_1, S_2, \cdots, S_k\}$, surfaces belong to the place that Albot_1 believes that it is re-visiting.

Output: Confirmation whether two places are the same or not.

Begin

1: **for** all S_j **do**
2: **for** all S_k **do**
3: Assume $S_j = S_k$.
4: Matched_Surfaces = 0;
5: **for** all S_k **do**
6: Transform S_k onto the metric map of P_c centering on $S_j, P'_{vs} \leftarrow S'_k$.
7: **end for**
8: **if** S'_k and S_j are matched, i.e. they roughly co-locate and have the same orientation **then**
9: Matched_Surfaces++;
10: **end if**
11: **if** Matched_Surfaces > 4 and they are located on both sides of the robot's current position **then**
12: Confirm that the two places are the same.
13: Break;
14: **end if**
15: **end for**
16: **end for**

End

4.2.1 Experiment 1: Learning a Network of Places

This experiment is designed to test whether Albot_1 can recognize that it is re-visiting a place (after travelling about 90 meters) and create a correct network of places. Figure 4.1 shows the loop through the environment used for this experiment. En indicates the exits recognized by Albot_1. Albot_1 starts and ends at point S, traveling in a clockwise direction. It went through 6 exits before re-visiting the starting place and should therefore compute a network of 6 unique places. Figure 4.2 shows the result of the network of places computed. Note that the metric map for each place shows some surfaces which are also part of an adjacent place (e.g. Surfaces in the dashed oval of P_1 are part of P_2) to which the robot has visited. These surfaces are seen from the exit prior to leaving the place.

Using the same route, the experiment is repeated with two different start-end positions (marked A and B in figure 4.1). In both cases, the robot successfully recognizes the place re-visited and creates a correct network of places (see figure 4.3).

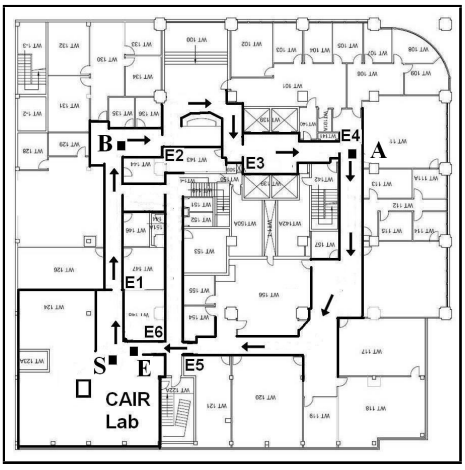


Figure 4.1: A single loop.

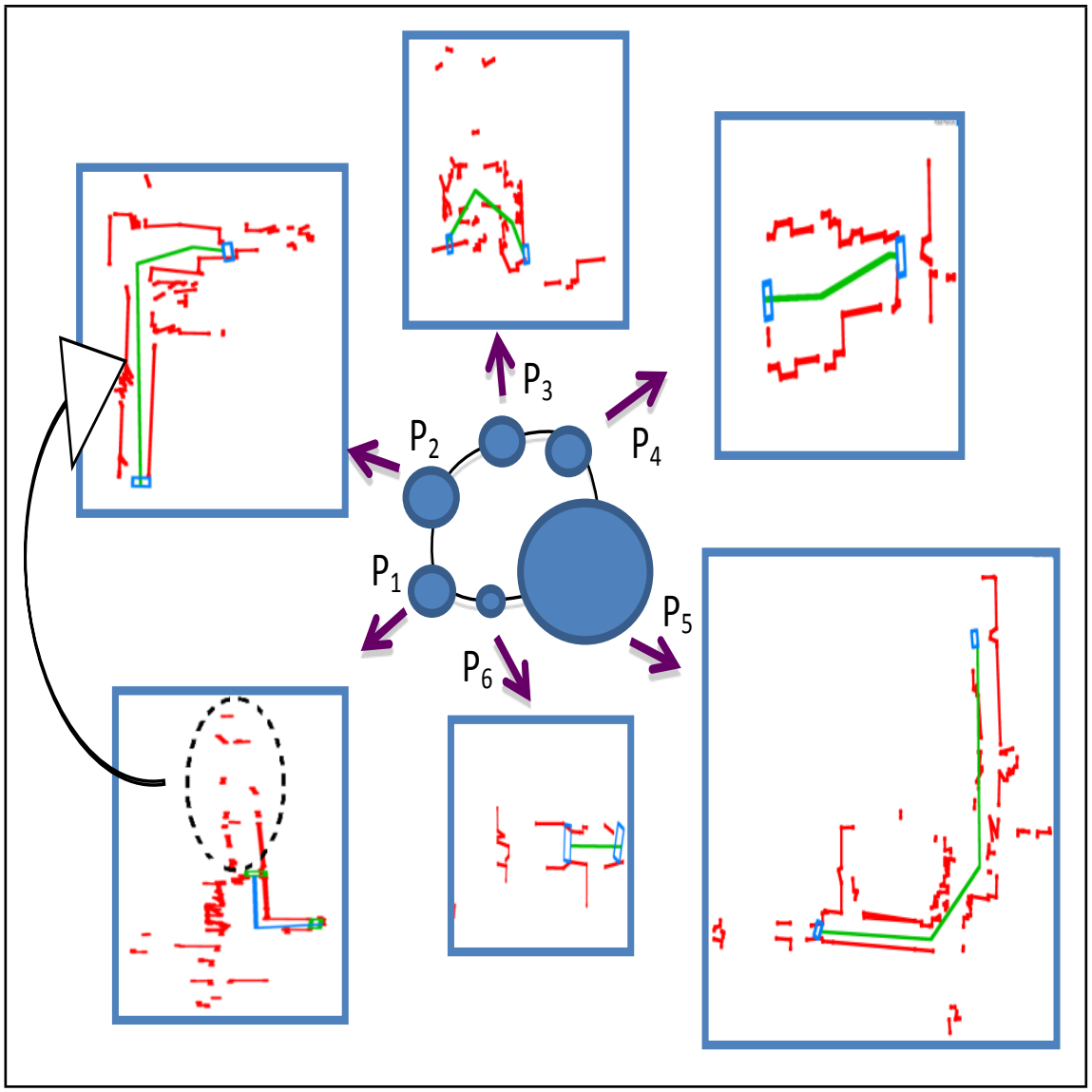
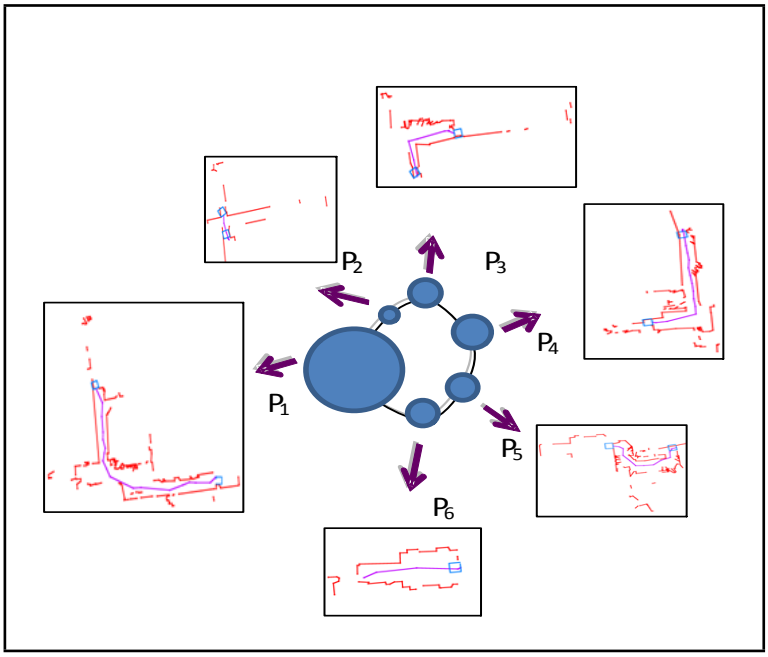
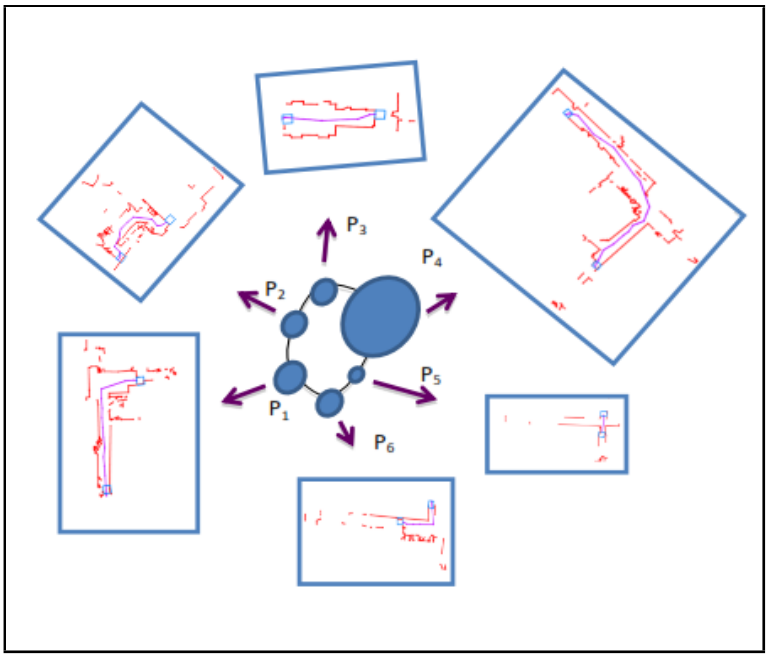


Figure 4.2: An example of a network of places computed. The circles represent the places learned (its relative size indicates the relative size of each place) and beside it, its metric map is displayed. The dotted line on the metric map of P_1 shows surfaces belonging to P_2 but seen from P_1 when it was first learned.



(a)



(b)

Figure 4.3: The network of place representation computed when Albot$_1$ started traveling from (a) point A and (b) point B as shown in figure 4.1.

4.2.2 Experiment 2: Recognizing Familiar Places

This experiment is designed to present a situation where Albot_1 fails to recognize that it is entering a familiar place. The new route traversed (about 150 meters) is as shown in figure. 4.4. As in the first experiment, Albot_1 returns to its starting position (i.e. CAIR Lab) however, as it enters that place, it is instructed to turn left to explore a part of the laboratory that it has not seen before. At the entrance, it detects the surfaces marked P_1 but there are not enough of them to trigger a check on whether it has indeed returned to P_1. Consequently, it thinks it is in a new place and continues its exploration. When Albot_1 returns to the exit, E6, where it came from, it creates a new node, P_7, in its network of place representation as shown in figure 4.5a. From the exit information, it knows that it is entering P_6. It turns left and discovers more of P_6 and it later exits into a new place, P_8. When it crosses from P_8, to P_2 via E8, it recognizes P_2 as shown in figure 4.5b. Note that P_2 is seen from a very different perspective. From here, it turns left, crosses the exit E1, and recognizes that it is moving into P_1 via the exit used. While exploring P_1, it detects some surfaces belonging to P_7 and confirms that it is (figure 4.5c). Figure 4.6 shows the updated network of place representation as a result of traversing the route as shown in figure 4.4. Note that the metric maps of places P_1 and P_6 are updated with more information and a description for P_8 is added.

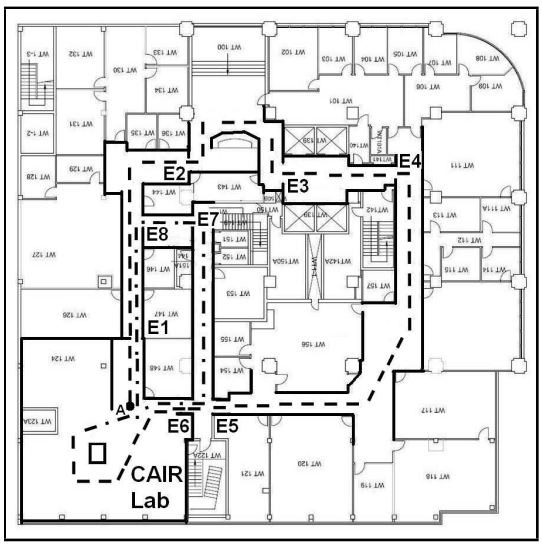
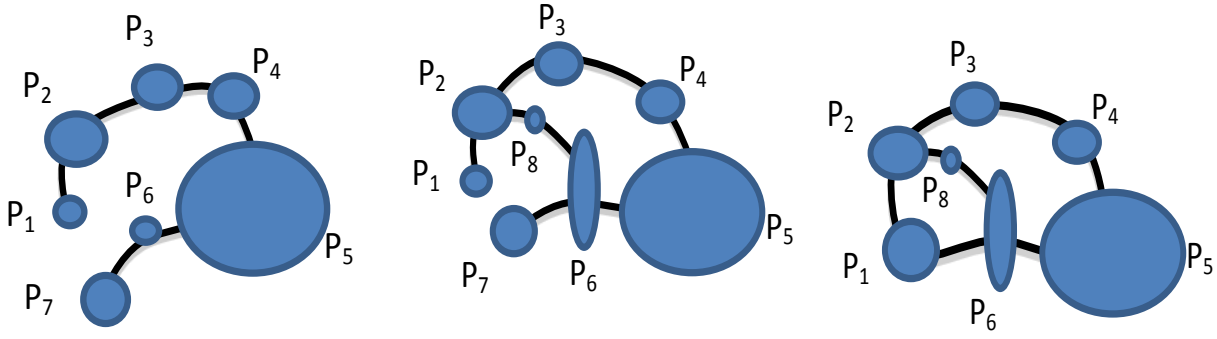


Figure 4.4: A double loop through the test environment.



(a) (b) (c)

Figure 4.5: (a) (b) (c) The network of place representation learned at 3 different stages during the experiment.

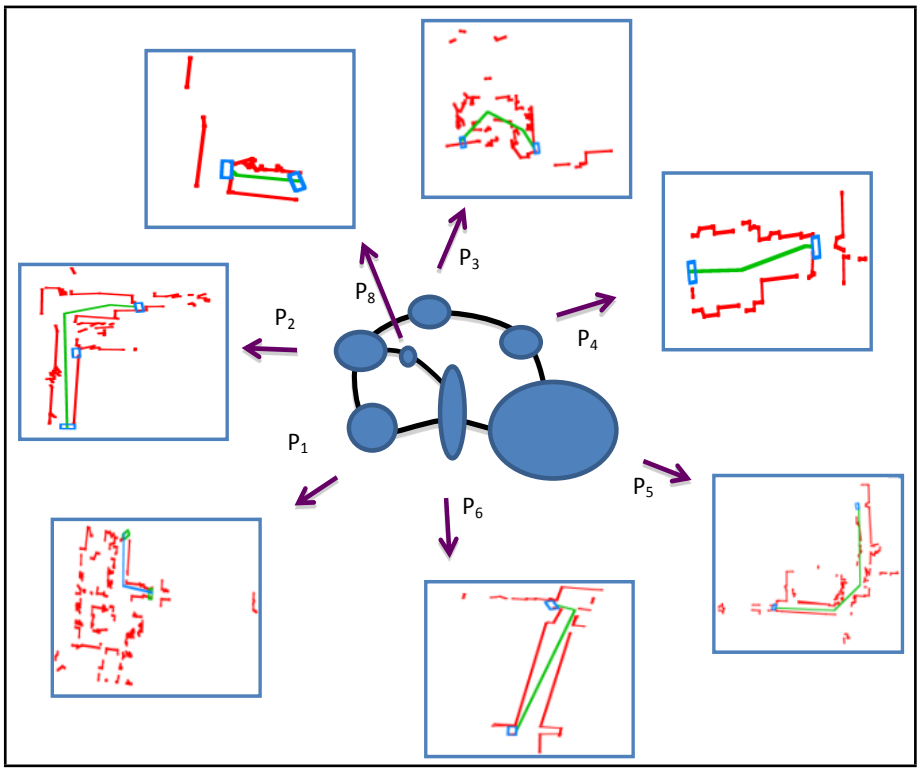


Figure 4.6: The network of place representation after traversing the path as shown in figure 4.3a.

4.2.3 Experiment 3: Navigation

This experiment is designed to demonstrate how the network of places computed is used by Albot$_1$ for navigating in its environment. Many researchers have demonstrated the use of such a network for path planning (e.g. (Thrun, 1998)). However, since loop closing is not done at the level of the metric maps, place maps, retrieved from the network, are not aligned properly with its physical counterpart. Therefore, they cannot be used directly for local navigation. To overcome this problem, we bring these maps into the perceptual map, one at a time and as one moves from one place

to another. These maps can then be aligned locally at their common exit. Doing so enables the robot to know what lies immediately ahead and thus allows the robot to use the information to guide its local navigation. It is not necessary to bring all the details of a metric map of a place into the perceptual map especially if one is simply walking through the environment. For the latter, it suffices to know the whereabouts of the local landmarks and/or exits nearby.

To test the above idea, the two algorithms above (one for remembering places after crossing the exit and the other for confirming a place re-visited) are modified so that whenever a place is re-visited, its known exits are also added to the perceptual map. After learning the environment as in Experiment 1, the robot repeats the same route through the environment, re-visiting all the known places. If the metric map of a place is aligned properly locally (in this case, the exits only), the robot should find an exit in its perceptual map that co-locates with the physical exit encountered, thus knowing its whereabouts and which place it is visiting next. The two algorithms are modified (at step 13 in bold) as shown on the next page.

When adding exits of P_v to the perceptual map (step 13), one uses the exit just crossed as a reference point for calculating the coordinates of the exits of P_v in the perceptual map. These exits are stored in a separate list so that they could be easily accessed (for step 13).

This experiment begins with Albot_1 traversing the same route as that shown in figure 4.1. When Albot_1 has confirmed that it is re-visiting P_1 (i.e. CAIR Lab.), it adds the exits of P_1 (in this case, just E1) to its perceptual map. Such exits (retrieved from its network of places, but is aligned locally) are shown as a light open rectangle in figure 4.7. Positions of actual exits used while traversing the environment are marked

Algorithm 7 Remembering Places Visited, remember Place $(\cdot)$

Input: (i) Perceptual map, $PM = \{S_1, S_2, \cdots, S_i\}$, where surfaces are marked as belonging to P_c when it is updated with new surfaces.

(ii) Crossed exit, $E_x = \{P_1(x, y), P_2(x, y)\}$, where the exit is.

(iii) Current place, $P_c = (\{\}, \{E_e\})$, where, $\{\}$ is an empty list of surfaces, and E_e is the exit through which Albot_1 came in this current place.

Output: Current place, $P_c = (\{S_1, S_2, \cdots, S_j\}, \{E_e, E_x\})$, where, $\{S_j\}$ is the list of j number of surfaces, E_e is the exit through which Albot_1 came to this current place, and E_x is the exit Albot_1 just crossed.

Begin

if an exit is crossed **then**
  for all S_i **do**
    if S_i is marked as a current place object **then**
      Add S_i to current place surface list, $P_c \leftarrow$ add (Si).
    end if
  end for
  Add E_x to current place exit list, $P_c \leftarrow$ add (E_x).
  if P_c is a new entry in the network of places **then**
    Create a new node in the map for P_c.
  else (update place)
    Combine its description with the appropriate entry in the network of place representation.
    Update the exit information between this place and the previous place so that it makes a correct reference to this place node;
    Check if the exit just crossed is leading to a known place, **or that it co-locates with a known exit leading to,** say, P_v. If so, mark P_c as equivalent to P_v. **Add exits of P_v to the perceptual map (this is the alignment step).**
  end if
  Set current place to $P_c = (\{\}, \{\})$;
  Update perceptual map with the current view and update exit list of current place to include the exit just crossed as entering exit: $P_c = (\{\}, \{E_e\}) \leftarrow$ add (E_x).
end if

End

Algorithm 8 Recognizing Places Re-visited, recognize Place $(\cdot)$

Input: (i) Surfaces, $P_{cs} = \{S_1, S_2, \cdots, S_j\}$, surfaces which belong to current place;

(ii) Surfaces, $P_{vs} = \{S_1, S_2, \cdots, S_k\}$, surfaces belong to the place that Albot_1 believes that it is re-visiting.

Output: Confirmation whether two places are the same or not.

Begin

1: **for** all S_j **do**
2: **for** all S_k **do**
3: Assume $S_j = S_k$.
4: Matched_Surfaces = 0;
5: **for** all S_k **do**
6: Transform S_k onto the metric map of P_c centering on $S_j, P'_{vs} \leftarrow S'_k$.
7: **end for**
8: **if** S'_k and S_j are matched, i.e. they roughly co-locate and have the same orientation **then**
9: Matched_Surfaces++;
10: **end if**
11: **if** Matched_Surfaces > 4 and they are located on both sides of the robot's current position **then**
12: Confirm that the two places are the same.
13: **Add exits of P_v to the perceptual map.**
14: Break;
15: **end if**
16: **end for**
17: **end for**

End

using a square. If these two exits co-locate, then one knows that one is entering into a known place. In figure 4.7, we find the two do in fact co-locate and thus when the robot crosses E1, it knows that it is re-visiting P_2 and immediate aligns its exit, E2, in its perceptual map. This process continues until the robot reaches P_6. In figure 4.7, we also show the original position of the exits retrieved from the network of place representation when they were learned. These exits are marked using a filled rectangle and as can be seen in figure 4.7, they would have shown the wrong whereabouts of these exits. Figure 4.8 shows the midpoint distance of these exits from the exits perceived in the perceptual map

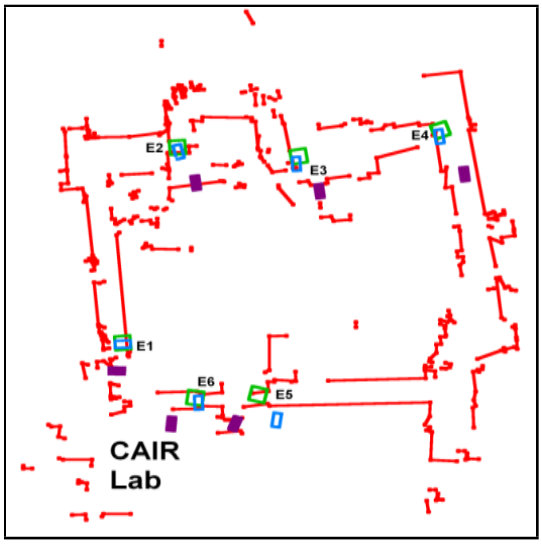


Figure 4.7: Predicting the whereabouts of exits when re-visiting a place.

4.3 Discussion

The idea that the perceptual map is an inexact map turns out not to be problematic when computing a network of places from it. It allows Albot_1 to automatically learn

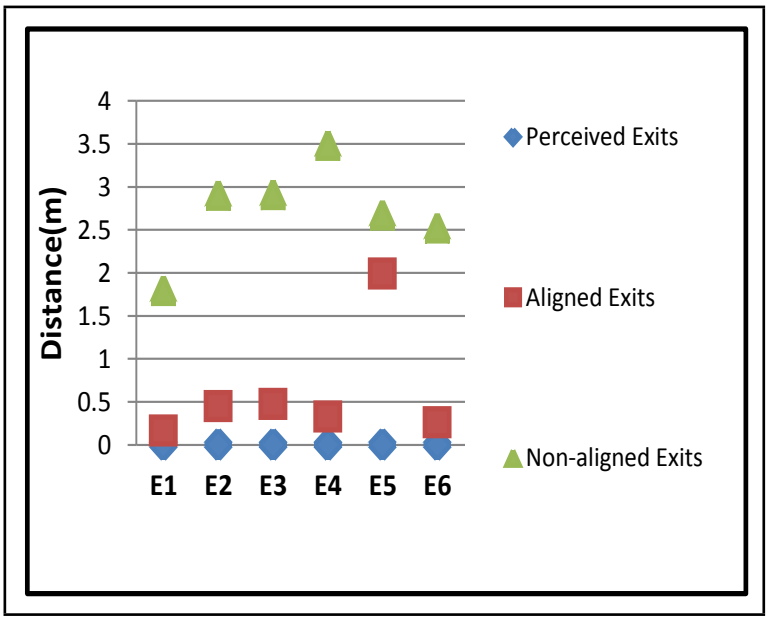


Figure 4.8: A comparison of midpoints distances between exits predicted (square denotes aligned and triangle denotes non-aligned) and exits actually perceived.

places that are not of a fixed structure. In fact, learned this way, the shape of a place can be complex (as shown in P_5 which is a large irregular shaped corridor), its boundary can overlap with other places and its shape can be incomplete. If Albot_1 were able to identify what it perceives and/or reason about a place, its representation could be enriched. When recognition of a place fails, it does not necessarily have disastrous consequences, as is the case in current robotic systems. Furthermore, having a place representation that does not have a pre-determined shape allows one to use a variety of heuristics for its recognition; similar to the way in which, humans and animals are observed to make use of a rich variety of information to recognize places. When travelling in a familiar environment, a network of places is known to be useful for planning a path between known places. However, if the metric map were not exact, wouldn't the execution of the plan pose a problem? Using Albot_1, we show

that this is not the case except when the place is large . We show how the local maps can provide expectations of what lies ahead. If the place is a large space (e.g. P_5 in Experiment 3), the predictions (e.g. Exit E5) can be distorted initially, but one's representation could improve over time.

Chapter 5

Conclusion

I have successfully implemented Albot$_1$, a robot designed to investigate how an autonomous agent computes an imprecise and incomplete map. This chapter concludes the thesis by discussing the significance of the results obtained, limitations and possible future works.

The Perceptual Map

There is strong evidence that some form of a map that is imprecise and incomplete exists in our head. A simple "proof" is our ability to point to unseen locations. Yet, such a map is little understood and it leads to serious debates among psychologists (see chapter 2). With Albot$_1$, I show how an incomplete and imprecise map can be computed. Furthermore, I show that such a map both dynamic and transient in nature. It is dynamic because the map keeps expanding as one moves out of the current local space into a new local space, despite the fact that one might be re-visiting a part of the environment. It is transient because information in it can be over-written by the incoming information or deleted because it has become obsolete. Having these two properties suggest that the map computed is not like a cartographic map of the

environment but rather a trace map of its journey through the environment.

Albot$_1$'s map turns out to be quite accurate as shown in the orientation experiment (Experiment 2). Albot$_1$ orients better than humans. This situation arises because Albot$_1$, using a laser sensor, has much more accurate distance information and Albot$_1$ also updates its map more frequently. The latter is due to Albot$_1$'s choice of landmarks. Albot$_1$'s landmarks are surfaces with an occluding point and such surfaces are often perceived close to Albot$_1$. For humans, a landmark could be a gigantic tower at a distance. Interestingly, Albot$_1$ could easily find itself in situations where it could not perceive any landmark (Experiment 1). Albot$_1$ overcomes this problem by resorting to a more "primitive" process of path integration. It switches from tracking landmarks to performing path integrations and this highlights that other mechanisms (and other cues in addition to landmarks) are needed when computing one's perceptual map.

The Process

The traditional process for computing one's map is to transform each successive view to form an integrated map. Such a process requires constant tracking of one's position in space, constant updating of one's map, and constant correction of errors caused by one's sensors. None of these requirements are needed in Albot$_1$'s process. Instead it tracks landmarks across successive views and updates its map when it enters another local environment. This creates a process that is not data intensive and does not require much updating (see Experiments 4 and 5). In this sense, Albot$_1$'s process is simpler and more cognitively relevant. Tracking landmarks is something that is commonly observed in human's spatial behavior. Furthermore, it is hard to imagine the

requirement that humans be constantly aware of their exact position in an absolute map.

The Cognitive Map

Albot_1 does not recognize places at the perceptual level rather it moves and adds new information to its map and deletes any overlapping information. Therefore, one question arises immediately: Can Albot_1 use such a map to perform cognitive related tasks, such as navigation through familiar places. If not, Albot_1's map will always be in a state of flux. To solve this problem, Albot_1 computes a more enduring representation of its environment from the transient perceptual map. It uses its transient map to familiar locations and the enduring map to find a route to reach its destination. Interestingly, psychologists support the idea of having multiple representations for dealing with different problems. They describe a two-stage mapping process whereby one first computes a transient map and then a more enduring cognitive map. Following the idea that the latter is computed via some abstraction process, I implemented one such process and show Albot_1 can compute a network of places from its perceptual map.

The network of places that Albot_1 computed shares many of the characteristics observed in the human system. For example, places could have fuzzy boundaries, overlapping boundaries, ill-defined shape, etc. Unlike the more traditional robotics approach, failure to recognize a place does not disrupt the process significantly; one might be lost but not destroyed. One interesting result is a demonstration of how the enduring information is projected onto the perceptual map to aid one's exploration of a familiar place.

On Robot Mapping

A new mapping process has been developed and tested on a mobile robot equipped with a laser sensor and an odometer. While the development of the process is to model an interesting aspect of cognitive mapping (i.e one that computes an inexact and incomplete map), it nonetheless provides a novel process for robot mapping and has some advantages over the existing popular SLAM-based paradigm for robot mapping (see Experiment 5). These advantages include:

1. Using less views to compute the map;
2. No integration of incoming view with information in the map; and
3. No error corrections.

It thus offers an interesting alternative paradigm for robot mapping and in particular for mapping in very large environments. Furthermore, it is expected that future robots would require huge computing power to perform various tasks, thus having a less demanding mapping process (in terms of computational resources) would be an advantage.

On Yeap's Theory of Perceptual Mapping

The implementation of Yeap's theory of mapping on a robot equipped with a laser sensor and an odometer requires the development of algorithms for the robot to track landmarks in successive views. Apart from this algorithm, no changes were made to Yeap's original theory and the successful implementation of it provides confirmatory

evidence that the process is both robust and useful. The implementation follows closely the three key steps according to Yeap's theory, namely:

1. Views remembered denote local environments visited;
2. Local features are tracked for localization; and
3. The map does not need to be updated continuously.

However, the implementation also reviews three interesting aspects of such a process, namely:

1. New views are always added to the map as they are without integrating with what is known in the map;
2. The perceptual map computed does not inform whether one is re-visiting a part of the environment or not; and
3. The map computed is unlike a cartographic map; it is more like a trace map of one's journey through the environment.

It is surprising that one is unable to recognize where one is using the perceptual map computed. It is argued that this latter ability should come from a separate recognition process.

Yeap's theory did not describe how such a recognition process would work in tandem with the mapping process and in particular how one would recognize that one is returning to a familiar part of the environment. I extended the theory by showing how an abstract representation (in the form of a network of places) could be learned from the dynamic map computed and using it to recognize places re-visited. I also

demonstrated one important use of a cognitive map, the ability to predict what lies ahead (see Chapter 4).

Limitations

Current implementation of the theory has three serious limitations:

1. The implementation did not investigate in depth the deletion of the map computed when new information is added to the map. This is an important step as it affects what is to be deleted and this in turn affects the usefulness of the map remained in memory.

2. The testing of the algorithm did not include more complex environments such as outdoor environments and/or cluttered indoor environments. Such testing might help illuminate unexpected problems in the implementation.

3. While the implementation shows a network of places could be learned, it relies on information in its perceptual map to detect re-visiting of a part of the environment. This is a severe limitation given that information in it could be deleted prior to re-visiting.

Future Work

While this study provides significant insights into the underlying process supported by Yeap's theory of perceptual mapping, it also highlights many significant questions that need to be explored in the future. Some of these questions are outlined below:

1. Albot_1's map is reasonably accurate. Could this be due to the laser sensor providing accurate distance information? More importantly, could we still get a useful map if the sensor is as illusory as that found in human vision? One

important next step is to implement Albot_2, a robot equipped with human-like vision. Researchers would need to modify the way landmarks are detected in subsequent views and develop a novel method for localizing itself in such a situation.

2. Albot_1 could be turned into a useful practical robot by improving upon its existing algorithms, making it more versatile and able to cope with a more complex environment (such as outdoor environments). For example, one could develop better algorithms for recognizing landmarks in subsequent views and develop algorithms for the robot to identify familiar parts of the environment.

3. The algorithm for removing parts of the map before adding a new view has not been properly investigated. Future work could investigate different algorithms for doing so and study their effects on the map.

4. Processes that utilize such a perceptual map for recognition, planning, and creating an enduring cognitive map have not been studied in depth and future work should look at extending existing work beyond the computation of a perceptual map.

5. Empirical data in support of the current process has not been provided and future work should look at reviewing such data in light of the new process developed.

References

Allen, G. L., Kirasic, K. C., Siegel, A. W., & Herman, J. F. (1979). Developmental issues in cognitive mapping: the selection and utilization of environmental landmarks. *Child Development*, *50*(4), 1062-1070.

Barrera, A., & Weitzenfeld, A. (2008). Biologically-inspired robot spatial cognition based on rat neurophysiological studies. *Autonomous Robot*, *25*, 147-169.

Bar-Shalom, Y., & Fortmann, T. E. (1988). *Tracking and data association.* Academic Press.

Beeson, P., Modayil, J., & Kuipers, B. J. (2010). Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *The International Journal of Robotics Research*, *29*(4), 428-459.

Bennett, A. T. D. (1996). Do animals have cognitive maps? *The Journal of Experimental Biology*, *199*, 219-224.

Blanco, J., Fernandez-Madrigal, J., & Gonzalez, J. (2008). Toward a unified bayesian approach to hybrid metric-topological slam. *IEEE Transactions on Robotics*, *24*(2), 259-270.

Borges, G. A., & Aldon, M. J. (2004). Line extraction in 2D range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, *40*, 267-297.

Burgard, W., Stachniss, C., Grisetti, G., Steder, B., Kummerle, R., Dornhege, C., ... Tardos, J. D. (2009). A comparison of slam algorithms based on a graph of relations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (p. 2089 - 2095). St. Louis, USA.

Burgess, N. (2006, December). Spatial memory: how egocentric and allocentric combine. *Trends in Cognitive Neuroscience*, *10*(12), 551-557.

Carr, S., & Schissler, D. (1969). The city as a trip: Perceptual selection and memory in the view from the road. *Environment and Behaviour*, *1*, 7-35.

Cartwright, B. A., & Collet, T. S. (1982, February). How honey bees use landmarks to guide their return to a food source. *Nature*, *295*, 560 - 564.

Cheng, K. (1986). A purely geometric module in the rat's spatial representation. *Cognition*, *23*, 162-171.

Cheng, K., & Newcombe, N. S. (2005). Is there a geometric module for spatial reorientation? squaring theory and evidence. *Psychonomic Bulletin and Review*, *12*, 1-23.

Chown, E., Kaplan, S., & Kortenkamp, D. (1995). Prototypes, location, and associative networks (plan): Towards a unified theory of cognitive mapping. *Cognitive Science*, *19*(1), 1-51.

Cornell, E. H., Heth, C. D., & Alberts, D. M. (1994, November). Place recognition and way finding by children and adults. *Memory and Cognition*, *22*(6), 633 - 643.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1052-1067.

Downs, R. M., & Stea, D. (1973). *Image and environment: Cognitive mapping and spatial behaviour*. Chicago, IL: Aldine Press.

Durrant-Whyte, H., & Bailey, T. (2006, June). Simultaneous localization and mapping: Part i. *IEEE Robotics & Automation Magazine*, *13*(2), 99-110.

Dyer, F. C. (1991). Bees acquire route-based memories but not cognitive maps in a familiar landscape. *Animal Behaviour*, *41*(2), 239-246.

Eliazar, A., & Parr, R. (2003). Dp-slam: fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (p. 1135-1142). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Fairfield, N., Kantor, G., & Wettergreen, D. (2007). Real-time slam with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, *24*(1/2), 3 - 21.

Fermuller, C., Cheong, L., & Aloimonos, Y. (1997, November). Visual space distortion. *Biological Cybernetics*, *77*(5), 323-337.

Fischler, M., & Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381-395.

Fox, D., Burgard, W., & Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*.

Frankenstein, J., Meilinger, T., Mohler, B. J., & Bulthoff, H. H. (2009). Spatial memory for highly familiar environments. In N. Taatgen & H. V. Rijn (Eds.), *Proceedings of the 31st Annual Conference of the Cognitive Science Society* (p. 2650-2655). Red Hook, NY: Curran.

Gallistel, C. R. (1990). *The organisation of learning*. Cambridge, MA: MIT Press.

Glennerster, A., Hansard, M. E., & Fitzgibbon, A. W. (2009). View-based approaches to spatial representation in human vision. In D. Cremers, B. Rosenhahn, A. L. Yuille, & F. R. Schmidt (Eds.), *Statistical and Geometrical Approaches to Visual Motion Analysis* (p. 193-208). Berlin: Springer-Verlag.

Golledge, R. G. (1999). Human wayfinding and cognitive maps. In R. G. Golledge

(Ed.), *Wayfinding behavior: Cognitive mapping and other spatial processe* (p. 5-45). Baltimore: John Hopkins University Press.

Gould, J. L. (1987, February). Landmark learning by honey bees. *Animal Behaviour*, *35*(1), 26-34.

Grisetti, G., Kmmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, *4*(2), 31-43.

Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, *23*, 34-46.

Hafner, V. V. (2008). Robots as tools for modeling navigation skills - a neural cognitive map approach. In E. Jefferies & W. K. Yeap (Eds.), *Robotics and cognitive approaches to spatial mapping* (p. 315 - 324). Berlin: Springer.

Hahnel, D., Burgard, W., Fox, D., & Thrun, S. (2003). An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (p. 206-211). Las Vegas, Nevad.

Hartley, T., Lever, C., Burgess, N., & O'Keefe, J. (2013). Space in the brain: How the hippocampal formation supports spatial cognition. *Philosophical Transactions of the Royal Society*, *B 369*.

Hirtle, S. C., & Jonides, J. (1985). Evidence of hierarchies in cognitive maps. *Memory and Cognition*, *13*(3), 208-217.

Hossain, M. Z., & Yeap, W. (2013). How albot1 computes its topological-metric map. *Procedia - Social and Behavioral Sciences*, *97*(0), 553 - 560. (The 9th

International Conference on Cognitive Science)

Huber, D. F., & Vandapel, N. (2006). Automatic three-dimensional underground mine mapping. *The International Journal of Robotics Research*, *25*(7), 7-17.

Hurlbert, A. C. (1994, May). Visual perception: Knowing is seeing. *Current Biology*, *4*(5), 423-426.

Ishikawa, T., & Montello, D. R. (2006). Spatial knowledge acquisition from direct experience in the environment: Individual differences in the development of metric knowledge and the integration of separately learned places. *Cognitive Psychology*, *52*, 93 - 129.

Jacobs, L. F., & Limant, E. R. (1991). Grey squirrels remember the locations of buried nuts. *Animal Behavior*, *41*, 103-110.

Jefferies, M. E., Baker, J., & Weng, W. (2008). Robot cognitive mapping - a role for a global metric map in a cognitive mapping process. In M. E. Jefferies & W. K. Yeap (Eds.), *Robotics and cognitive approaches to spatial mapping* (Vol. 38, p. 265-279). Berlin: Springer.

Jefferies, M. E., & Yeap, W.-K. (Eds.). (2008). *Robotics and cognitive approaches to spatial mapping* (Vol. 38). Berlin: Springer.

Kim, J., & Sukkarieh, S. (2007). Real-time implementation of airborne inertial-slam. *Robotics and Autonomous Systems*, *55*, 62-71.

Konolige, K., Marder-Eppstein, E., & Marthi, B. (2011, May). Navigation in hybrid metric-topological maps. In *IEEE International Conference on Robotics and Automation (ICRA 2011)* (p. 3041-3047). doi: 10.1109/ICRA.2011.5980074

Kretzschmar, H., Grisetti, G., & Stachniss, C. (2010). Lifelong map learning for graph-based slam in static environments. *Kunstliche Intelligenz*, *24*, 199-206.

Kuipers, B. (1978). Modeling spatial knowledge. *Cognitive Science*, *2*, 129-153.

Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, *119*, 191233.

Kuipers, B. (2001). The skeleton in the cognitive map: A computational hypothesis. In *Proceedings of the 3rd International Space Syntax Symposium* (Vol. 10, p. 1-10). Atlanta.

Kuipers, B., & Byun, Y. T. (1988). A robust qualitative method for spatial learning in unknown environments. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-88)* (p. 774-775). Los Altos, CA: Morgan Kaufman.

Kumaran, D., & Maguire, E. A. (2005, August). The human hippocampus: Cognitive maps or relational memory? *The Journal of Neuroscience*, *25*(31), 7254-7259.

Lambrinos, D., Mller, R., Labhart, T., Pfeifer, R., & Wehner, R. (2000). A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, *30*, 39-64.

Lee, S. A., & Vallortigara, E. S. S. G. (2012). Chicks, like children, spontaneously reorient by three-dimensional environmental geometry, not by image matching. *Biology Letters*, *8*(4), 492-4.

Lynch, K. (1960). *The image of the city.* Cambridge, MA: MIT Press.

Maury, D. L., Mauch, R. J., Hammer, A. N., & Bingman, V. P. (2010, September). Spatial and feature-based memory representation in free-flying homing pigeons. *Animal Cognition*, *13*(5), 733-743.

McNamara, T. P. (2003). How are the locations of objects in the environment represented in memory? In *Spatial cognition iii: Routes and navigation, human*

memory and learning, spatial representation and spatial reasoning (p. 174-191). Berlin: Springer-Verlag.

Milford, M., & Wyeth, G. (2007). Spatial mapping and map exploration: A bio-inspired engineering perspectives. In S. Winter, M. Duckham, L. Kulik, & B. Kuipers (Eds.), *the 8th International Joint Conference on Spatial Information Theory (COSIT)* (p. 203-221). Berlin: Springer Verlag.

Minguez, J., & Montano, L. (2005). Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robotics and Autonomous Systems*, *52*, 290-311.

Montemerlo, M., Roy, N., Thrun, S., Hahnel, D., Stachniss, C., & Glover, J. (2002). *Carmen - the Carnegie Mellon Robot Navigation Toolkit.* Retrieved from `http://carmen.sourceforge.net`

Montemerlo, M., & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using fastslam. In *IEEE International Conference on Robotics and Automation ICRA (2003)* (Vol. 2, p. 1985 - 1991).

Moore, G. T., & Golledge, R. G. (1976). *Environmental knowing: Theories, research, and methods.* Stroudsbury, PA: Dowden, Hutchinson & Ross.

Moravec, H. P. (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 61 - 74.

Mou, W., & McNamara, T. P. (2002, January). Intrinsic frames of reference in spatial memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *28*(1), 162-170.

Mou, W., McNamara, T. P., Valiquette, C. M., & Rump, B. (2004). Allocentric and egocentric updating of spatial memories. *Journal of Experimental Psychology:*

Learning, Memory, and Cognition, *30*(1), 142-157.

Nguyen, V., Martinelli, A., Nicola, & Siegwart, R. (2005). A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)* (p. 1929 - 1934). IEEE.

Nuchter, A., Lingemann, K., Hertzberg, J., & Surmann, H. (2007). 6D SLAM - 3D mapping outdoor environments. *Journal of Field Robotics*, *24*(8/9), 699-722.

O'Keefe, J., & Nadel, L. (1978). *The hippocampus as a cognitive map.* Oxford: Clarendon Press.

Olton, D. S. (1977). Spatial memory. *Scientific America*, *236*, 82-98.

Pathak, K., Pfingsthorn, M., Birk, A., Schwertfeger, S., Vaskevicius, N., & Poppinga, J. (2010). Online 3D SLAM by registration of large planar surface segments and closed form pose-graph relaxation. *Journal of Field Robotics*, *27*(1), 52-84.

Pfister, S. T., Roumeliotis, S. I., & Burdick, J. W. (2003). Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2003)* (p. 1304-1311). IEEE.

Poncela, A., Perez, E., Bandera, A., Urdiales, C., & Sandoval, F. (2002). Efficient integration of metric and topological maps for directed exploration of unknown environments. *Robotics and Autonomous Systems*, *41*, 21-39.

Presotto, A., & Izar, P. (2010, July). Spatial reference of black capuchin monkeys in brazilian atlantic forest: egocentric or allocentric? *Animal Behaviour*, *80*(1), 125-132.

Raudies, F. (2013). *Optic flow.* Retrieved from `http://www.scholarpedia.org/`

article/Optic_flow

Remolina, E., & Kuipers, B. J. (2004). Towards a general theory of topological maps. *Artificial Intelligence*, *152*, 47-104.

Roberts, W. A. (1979). Spatial memory in the rat on a hierarchical maze. *Learning and Motivation*, *10*, 117-140.

Rock, I. (1973). *Orientation and form.* New York: Academic Press.

Rosati, A. G., & Hare, B. (2012, November). Chimpanzees and bonobos exhibit emotional responses to decision outcomes. *Developmental Science*, *15*(6), 840-53.

Sholl, M. J. (2001). The role of a self-reference system in spatial navigation. In D. R. Montello (Ed.), *Spatial information theory: Foundations of geographical information science* (p. 217-232). Berlin: Springer-Verlag.

Siegel, A. W., & White, S. H. (1975). The development of spatial representations of large-scale environments. In W. Reese (Ed.), *Advances in child development and behavior* (Vol. 10, p. 9-55). New York: Academic Press.

Sovrano, V. A., Rigosi, E., & Vallortigara, G. (2012). Spatial reorientation by geometry in bumblebees. *PLoS One*, *7*(5).

Sovranoa, V. A., Bisazzaa, A., & Vallortigarab, G. (2002). Modularity and spatial reorientation in a simple mind: encoding of geometric and nongeometric properties of a spatial environment by fish. *Cognition*, *85*, 51-59.

Spetch, M. L., Cheng, K., MacDonald, S. E., & Linkenhoker, B. A. (1997, March). Use of landmark configuration in pigeons and humans: Ii. generality across search tasks. *Journal of Comparative Psychology*, *111*, 14-24.

Stachniss, C., Frese, U., & Grisetti, G. (2007). *Openslam.* Retrieved from http://

`openslam.org/`

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, *99*(1), 21-71.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics* (Vol. 1). The MIT Press.

Thrun, S., Fox, D., Burgard, W., & Dellaert, F. (2000). Robust monte carlo localization for mobile robots. *Artificial Intelligence*, *128*(1-2), 99-141.

Tolman, E. C. (1948, July). Cognitive maps in rats and men. *Psychological Review*, *55*(4), 189-208.

Vandorpe, J., Brussel, H. V., & Xu, H. (1996). Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1996)* (p. 901 - 908). IEEE.

Waller, D., & Hodgson, E. (2006, July). Transient and enduring spatial representations under disorientation and self-rotation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *32*(4), 867-882.

Wang, R. F., Crowell, J. A., Simons, D. J., Irwin, D. E., Kramer, A. F., Ambinder, M. S., ... Hsieh, B. B. (2006). Spatial updating relies on an egocentric representation of space: Effects of the number of objects. *Psychonomic Bulletin and Review*, *13*(2), 281 - 286.

Wang, R. F., & Spelke, E. S. (2000, December). Updating egocentric representations in human navigation. *Cognition*, *77*(3), 215-250.

Wang, R. F., & Spelke, E. S. (2002, September). Human spatial representation: Insights from animals. *Trends in Cognitive Sciences*, *6*(9), 376-382.

Wehner, R., & Menzel, R. (1990). Do insects have cognitive maps? *Annual Review of Neuroscience*, *13*, 403-414.

Wehner, R., Michel, B., & Antonsen, P. (1996). Visual navigation in insects: Coupling of egocentric and geocentric information. *Journal of Experimental Biology*, *199*, 129-140.

Wehner, R., & Wehner, S. (1990). Insect navigation: Use of maps or ariadnes thread? *Ethology, Ecology and Evolution*, *2*, 27-48.

Wong, C. K. (2008). *Cognitive inspired mapping by an autonomous mobile robot* (Unpublished doctoral dissertation). Computing and Mathematical Sciences, AUT.

Wulf, O., Arras, K. O., Christensen, H. I., & Wagner, B. (2004). 2D mapping of cluttered indoor environments by means of 3d perception. In *IEEE International Conference on Robotics and Automatlon* (p. 4204-4209). New Orleans, LA.

Wyeth, G., & Milford, M. (2009, September). Spatial cognition for robots - robot navigation from biological inspiration. *IEEE Robotics Automation Magazine*, *16*(3), 24-32. doi: 10.1109/MRA.2009.933620

Yeap, W. K. (1984). *Towards a computational theory of cognitive maps* (Unpublished doctoral dissertation). Department of Computer Science, University of Essex.

Yeap, W. K. (2011a). A computational theory of human perceptual mapping. In *Proceedings of the Cognitive Science Conference* (p. 429-434). Boston, USA: Cognitive Science Society.

Yeap, W. K. (2011b). How albot0 finds its way home: A novel approach to cognitive mapping using robots. *Topics in Cognitive Science*, *3*(4), 707–721.

Zoladek, L., & Roberts, W. A. (1978). The sensory basis of spatial memory in the

rat. *Journal of Animal Learning and Behaviour*, *6*(1), 77-81.