Localization of Mobile Robots in Indoor Environments Using Laser Range Data

By

Biswajit Sarkar

ENGG04200904001

Variable Energy Cyclotron Centre

A thesis submitted to
the Board of Studies in Engineering Sciences
in partial fulfillment of the requirements
for the Degree of

Doctor of Philosophy

of

Homi Bhabha National Institute

April, 2016

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by **Biswajit Sarkar** titled "**Localization of mobile robots in indoor environments using laser range data**" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

Chairman: Prof. D. K. Srivastava	Date:
Guide/Convener: Prof. Debranjan Sarkar	Date:
Examiner: Prof. K. Madhava Krishna	Date:
Member 1: Prof. Alok Chakrabarti	Date:
Member 2: Prof. Prabir K. Pal	Date:

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date:

Place:

Guide

Statement by Author

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at Homi Bhabha National Institute (HBNI) and is deposited in the Library to be made available to borrowers under rules of the HBNI.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the Competent Authority of HBNI when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

(Biswajit Sarkar)

Declaration

I, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

(Biswajit Sarkar)

List of publications arising from the thesis

Journals

1. "A novel method for computation of importance weights in Monte Carlo localization on line segment-based maps", **Biswajit Sarkar**, Surojit Saha and Prabir K. Pal, *Robotics and Autonomous Systems*, **2015**, *74*, Part A, 51–65.

2. "Building maps of indoor environments by merging line segments extracted from registered laser range scans", **Biswajit Sarkar**, Prabir K. Pal and Debranjan Sarkar, *Robotics and Autonomous Systems*, **2014**, *62*, 603–615.

3. "Radiation mapping inside the bunkers of medium energy accelerators using a robotic carrier", R. Ravishankar, T.K. Bhaumik, T. Bandyopadhyay, M. Purkait, S.C. Jena, S.K. Mishra, S. Sharma, V. Agashe, K. Datta, **B. Sarkar**, C. Datta, D. Sarkar and P.K. Pal, *Applied Radiation and Isotopes*, **2013**, *80*, 103–108.

4. "An efficient method for near-optimal polygonal optimization based on differential evolution", **Biswajit Sarkar**, *International Journal of Pattern Recognition and Artificial Intelligence*, **2008**, *22*, 1267–1281.

5. "A genetic algorithm-based approach for detection of significant vertices for polygonal approximation of digital curves", **Biswajit Sarkar**, Lokendra Kumar Singh and Debranjan Sarkar, *International Journal of Image and Graphics*, **2004**, *4*, 223–239.

6. "Hierarchical representation of digitized curves through dominant point detection", **Biswajit Sarkar**, Sanghamitra Roy and Debranjan Sarkar, *Pattern Recognition Letters*, **2003**, *24*, 2869–2882.

7. "Approximation of digital curves with line segments and circular arcs using genetic algorithms", **Biswajit Sarkar**, Lokendra Kumar Singh and Debranjan Sarkar, *Pattern Recognition Letters*, **2003**, *24*, 2585–2595.

Conferences

1. "Monte Carlo-based pose tracking on maps represented with line segments", Surojit Saha, **Biswajit Sarkar**, Prabir K. Pal, *Proceedings of the 2015 International Conference on Advances in Robotics, AIR–2015*, Goa, India, ACM, **2015**, 62:1–62:6.

2. "Localization of mobile robots in indoor environments", **Biswajit Sarkar**, Tanushyam Bhattacharjee, Prabir K. Pal and Debranjan Sarkar, *in: Proceeding of the Seminar on Applications of Computer & Embedded Technology (SACET)—2009, Kolkata, India*, **2009**, 73–75.

Others

1. "Localization in of mobile robots in indoor environments using laser range data, **Biswajit Sarkar**, *Doctoral Symposium, 2015 International Conference on Advances in Robotics (AIR–2015), organized by the Robotics Society of India*, Goa, India, July 5, 2015 (**Oral presentation**).

2. "Building maps of and localization in indoor environments using a laser range finder, **Biswajit Sarkar**, *International Workshop on Autonomous Vehicles and Mobile Robots*, Indian Institute of Technology Delhi, July 6–8, 2014 (**Invited talk**).

To

My Father

and

My Daughter

Acknowledgement

I would have not been in a position to even dream of writing this thesis had it not been for the painstaking effort of my father towards my education and upbringing in my formative years. His love for and confidence in me are my sources of constant inspiration.

I fondly recall the love and encouragement that I received from my teachers during my school and college days. They played a big part in shaping my future along with my mother and other members of our joint family.

It is matter of great pride and privilege for me to serve the Department of Atomic Energy (DAE). Starting from the initial days of my professional career in DAE, I was fortunate to have received invaluable mentoring from Prabir K. Pal and Debranjan Sarkar, and the present thesis is largely an outcome of that. I am thankful to the present and past Directors of Variable Energy Cyclotron Centre (VECC)—D. K. Srivastava, R. K. Bhandari and Bikash Sinha—for permitting me to commence and continue my research in the intellectually stimulating domain of mobile robot navigation. I am grateful to P. Barat, S. V. G. Ravindranath and M. Nasipuri for their kind words of encouragement and advice. Many of my colleagues in VECC, in general, and in Computer Division of VECC, in particular, had extended full cooperation and support in my research. Particular mention, in this context, must be made of Surojit Saha. Paramita Mukherjee, Dean-Academic (Engineering Sciences) at VECC has always had warm words of encouragement for me in my endeavour. I am particularly thankful to the staff members of the Scientific Information Resources Facility of VECC for their cooperation and support.

I would like to thank Steffen Gutmann for sharing his source codes with us many years ago, some of which have been used in the work reported in this thesis. I would also like to thank Francesco Amigoni and Rolf Lakaemper for providing source codes of their methods and other insightful suggestions. I deeply appreciate the valuable comments received from the anonymous reviewers of my journal papers. The public data, which have been used in my work, were obtained from the Robotics Data Set Repository (Radish). Thanks go to Nakju Lett Doh, Gian Maria Pelusi and Regis Vincent for providing the data.

Last but not the least, I would like to thank my wife for freeing me of many domestic reponsibilities and allowing me to procrastinate others so that I could devote extended hours and the weekends for my work. My dear daughter, Basabi, who is just too young to understand the essence of this thesis—it is just a "book" to her—was deprived of many play-hours because of my preoccupation. I have really missed her company during this period!

Contents

Synopsis

A necessary prerequisite for autonomous navigation of a mobile robot is the capability of *localization*, which continually provides the robot with a reliable estimate of its pose in a world coordinate system. If the spatial locations of the objects in the vicinity of the robot are known, and the robot is able to perceive its distance from these objects using, say, a Laser Range Finder (LRF), it is possible to make an estimate of its pose. Thus, it is essential to have an accurate model of the spatial arrangement of objects in the environment of the robot in the form of a *map*. But the process of automatically building a map necessitates that the robot is well aware of its pose while it moves around and perceives its environment. In the absence of map and exact pose information, the robot is required to simultaneously estimate both the map and localize itself relative to the (possibly partial) map built thus far—a problem known in the robotics community as the problem of *Simultaneous Localization and Mapping* (SLAM).

In many real-life applications, a robot is required to navigate continuously in environments that remain practically invariant over long periods of time. For all these applications, it is more appropriate to build a map of the working environment first and then use it for localization, rather than operate the robot in SLAM mode, which is complex, computation-intensive and less accurate. By decoupling mapping from localization, the robot can also be absolved of the job of map building while it is performing its assigned duty. These appliations require that a method is available for offline building of maps from range data collected by the robot during an initial phase when it is manually driven or tele-operated in its environment. A robust and accurate method for localization on maps built by such a method is also a prerequisite for the robot to move around safely and efficiently in the environment.

With the objective of achieving robust and accurate localization of mobile robots equipped with a LRF in (unmodified) indoor environments that practically remains invariant over extended periods of time, in this thesis, we propose

- an offline method for building maps of indoor environments by merging line segments extracted from registered laser range scans; and
- a robust and accurate method for localization on such maps based on the Monte Carlo framework.

Line segments are a natural choice in the representation of indoor environments. Moreover, maps based on line segments are compact, provide floating-point resolution and scale well with the environment size. The proposed method for map building first extracts line segments from individual laser ranges scans, which have been pre-registered. The extracted line segments are then organized in a tree-like hierarchy using two steps of density-based clustering. Each of the leaf nodes in the tree hold line

segments that generally arise from the same planar surface of the environment. All such line segments are finally merged to yield a resulting line segment that captures the attributes (viz., orientation, length and position) of the corresponding planar surface of the environment. The collection of all such resulting line segments defines a line segment-based map of the environment.

The proposed method has been successful in accurately building maps of large environments from datasets available in the public domain as well as from simulated and real-world data. Experimental results show that maps produced by the proposed method are generally better than those produced by two other methods reported in the literature in terms of the compactness of the maps and the lengths of the map segments. We also propose simple ways of quantitatively assessing the goodness of a line segment-based map produced in relation to the ground truth. To the best of our knowledge no effort towards such assessments was made in the past.

Monte Carlo Localization (MCL) is a powerful and popular approach for mobile robot localization. The benefits associated with the use of line segment-based maps and the effectiveness of MCL are sufficient motivations for implementing MCL on such maps. But Monte Carlo localization has seldom been studied in the context of line segment-based maps. A key step of the approach—and one that can endow it with or rob it of the attributes of accuracy, robustness and efficiency—is the computation of the so called importance weight associated with each hypothesized pose or particle.

In this thesis, we also propose a novel, heuristic-driven approach for the computation of importance weights in MCL on maps represented with line segments, and extensively study its performance in pose tracking. We also compare our method with three other methods reported in the literature. The comparative study, conducted using both simulated and real data, on maps built from real data available in the public domain clearly establish that the proposed method is more accurate, robust and efficient than the other methods.

In the concluding part of the thesis, we point out several open issues for further research. In the backdrop of what has been presented in the earlier part of the thesis, we discuss on a probable way to achieve added robustness and accuracy in localization so as to make the proposed methods of map building and localization more useful and efficacious for deployment in real-life applications.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Robots and autonomy

The word "robot" was first introduced in and popularized by the play R.U.R. (Rossum's Universal Robots) [1]. The play, penned by Karel Čapek, a Czech novelist, was first performed in 1921. In the play, robots were assembled from biological parts in a factory and served humans initially, but turned rebellious eventually. The notion of "robots" was further popularized by film-makers and science fiction writers, notable among them being Isaac Asimov, who introduced the term "robotics" [2].

Transending the domain of fiction and imagination, robots became a reality in the 1960s with their large-scale deployment for industrial automation. These robots took the form of robot arms—often referred to as *industrial manipulators*—that could be pre-programmed to perform repetitive tasks with precision and efficiency for extended periods of time in *structured* environments. The inability of these robots to perceive their environments and to act accordingly restricted their usage to well-defined tasks in structured environments only. Soon after, mobile robots in the form of *Automated Guided Vehicles* (AGVs) that followed pre-defined paths began to be used for efficient and large-scale material handling in factory environments. These robots, too, were deployed in structured settings because of their inability to contend with unanticipated situations that arose in real-worlds.

With advances in sensor technology and availability of high processing power at affordable costs, robots equipped with a suite of accurate and fast-acting sensors and reasonable computing power became commonplace since the 1990s. Coupled with these, advances in software technology [3] ensured that robots could now be endowed with the wherewithal necessary to make reasonable and consistent decisions in complex real-world environments based on uncertain and partially available information. This opened up the possibility of designing autonomous robots [4] that are capable of operating in real and unstructured environments for sufficiently long periods of time without

any external intervention. But, designing robots that could function in the desired way in all conceivable real-world situations is a tall order. So, autonomous robots, in common parlance, connote robots capable of autonomous operation in respect of a predefined task and environment.

Autonomous operation of a team of robots playing soccer can be witnessed in the RoboCup competitions [5]. Rovers with autonomous capabilities have been used to explore the surface of Mars by navigating on unknown and uncharacterized terrains [6]. The winning of the DARPA Grand Challenge [7] in 2005 by the robot Stanley [8] and successful completion of the course by its other competitors was vindication that autonomous driving in unstructured off-road environments had matured as a technology [9]. In 2007, the robot Boss won the DARPA Urban Challenge [10, 11] proving that it was capable of autonomous urban driving. Autonomous driverless cars on urban roads became a reality with the Google driverless car.

Autonomous robots, which share space with humans, are fast becoming a reality with their large scale deployment in homes, offices, hospitals and public places. A majority of these robots are wheeled and rely on autonomous navigation for their operation. Thus, it is useful and desirable to study issues and capabilities central to autonomous navigation of wheeled mobile robots. In this thesis, we study the problem of localization and map building insofar as they relate to autonomous navigation of mobile robots in indoor environments that have not been tailored (through the placement of reflectors or guidewires, for instance) to simplify the task of localization.

1.2 Mobile robots

Mobile robots are capable of moving around in their environments for performing their assigned work. They find application whenever the need for inspection, material transfer or object manipulation arise in hazardous and inhospitable areas where it is not practicable to depute human beings. These include, for instance, emergency response in nuclear disaster [12] and other nuclear-related services [13–16], inspecting and repairing leakage in pipelines [17–19] and removal and disposal of ammunition boxes [20]. Advances in field robotics are gradually making way for deployment of mobile robots in natural settings where the environments are dynamic and unstructured. For instance, space robots are being used for exploring the surface of the moon and the Mars [6,21–24]. Unmanned aerial vehicles are used as security robots for reconnaissance and surveillance in battle fields [25,26] and detection of forest fire [27–29]. Underwater robots are used for marine geological survey and repair of ships on sea [30, 31]. To minimize the dependence on manual labor, agricultural robots are used for harvesting, pruning and weeding [32–34]. Mobile robots are also used in forests for felling and subsequent transportation of trees. They are used in factory floors for large scale

transportation of materials [35–37], in hospitals for transportation of pharmaceuticals, meals and medical records [38,39] and in offices for delivery of mails [40–42]. Domestic robots are engaged in doing the daily chores at home like cleaning and assisting the aged [43–49]. Mobile robots also take people on various rides and on tours in museums for entertainment and amusement [50–53]. These are of course instances of only a handful of mobile robot applications; many other interesting and non-trivial applications abound in the literature and on the Web.

The capability of ground locomotion in mobile robots commonly arises from the use of wheeled mechanisms or articulated legs. Wheeled locomotion, though simple and efficient, is more suited for flat and hard terrains. A variant of wheeled locomotion is the tracked locomotion, which provides more surface contact and grip to the ground. They have good maneuverability and efficiency on rough and loose terrain but extremely inefficient otherwise. On soft and irregular terrains, legged locomotion is preferable as it makes only point contacts with the ground, although coordinated control of leg motions is in general fairly complex. Almost all indoor applications of mobile robots employ wheeled locomotion as the terrains therein are engineered to be smooth and hard.

A mobile robot must be able to perceive its environment if it is to navigate autonomously. The process of perception involves extraction of meaningful information from raw sensory data and subsequent formation of an effective internal *representation* or *model* out of the data. The robot can then use the internal representation to infer its relationship with the environment. A mobile robot is commonly equipped with a suite of sensors [54]. For sensing the extent of free space around itself, a mobile robot usually employs an ultrasonic range sensor. Typically operating at a frequency of 40 kHz and utilizing the principle of time-of-flight, ultrasonic sensors provide the range of nearby objects lying within a fixed solid angle. Although it provides poor directional information, it is useful in obstacle avoidance. It has a limited range (a few meters or less) and poor range resolution (a few cm). A *Laser Range Finder* (LRF) utilizes a laser beam to measure the range of the nearest object. Working on the principle of optical phase-shift, an LRF provides accurate range data (of the order of cm) at a very good angular resolution (half-a-degree or less). A mobile robot often employs a vision sensor (camera) to perceive its environment. But, vision sensors are generally limited in range and acuracy. In comparison, an LRF provides more accurate range data and is thus well suited for building an accurate spatial model of the environment (called *map*) and in locating the robot in the environment. An array of micro-switches, called *bumpers*, is almost always put all around the periphery of a mobile robot to sense its physical contact with any obstacle in the environment while on the move. The sensors referred to above are categorized as *exteroceptive* sensors inasmuch as they acquire information about the environment in which the robot is currently in. The *proprioceptive* sensors,

on the contrary, acquire information internal to the robot. Of particular importance is the *odometer*, which provides an estimate of the position and orientation of the robot by the integration of wheel rotations. However, odometers do not provide accurate estimates in the long run because of accumulation of errors with the length of travel. Nevertheless, they are useful for providing estimates of incremental changes in position and orientation. Accelerometers, gyroscopes and digital compasses are often used in conjunction with odometers to reduce errors in the odometry estimate.

1.3 Autonomous navigation of mobile robots

The capability of autonomous navigation permits a mobile robot to reach its goal from its current position, following a safe and reasonably efficient path, entirely on its own with no external intervention. The goal is either set by a human or is determined by the robot itself. The goal could be the ultimate destination of the robot or an intermediate waypoint on the path to it. The capability of autonomous navigation calls for cognitive ability on the part of the robot so that it is able to make "sensible" decisions for reaching the goal efficiently and reliably, based on its perceptions of the environment, which are more often partial, uncertain and noisy.

Before making a move, a robot must have with it a description of the path by traversing which it will move closer to its goal, if not reach the goal. It is generally possible to compute such a path (provided it exists) in advance, if the current position of the robot and that of the goal is known in addition to having a complete description of obstacles and free space (i.e., map) of the environment. The problem of computing such paths is generally discussed under the rubric of *path-planning* [55].

In the *Hierarchical Paradigm* of robot control [56], a robot first senses it environment and then based on this perception plans a path towards its goal while avoiding obstacles on the way. Subsequently, it acts so as to move along the planned path. After it has moved a little, it senses the world again, plans a path afresh and moves along it and the cycle of *Sense–Plan–Act* repeats (Figure 1.1a). Planning a path by taking into consideration the global arrangement of obstacles in the environment (called *global path planning*) is a compute-intensive process. Thus, frequent invocation of the Sense–Plan–Act cycle slows down navigation and is not practicable. But, inaccuracies in the robot actuators and the presence of dynamic obstacles in the environment necessitate that the planned paths are checked for their validity at frequent intervals and suitably modified, if required, so that the robot neither misses its goal nor hits an obstacle on the way. Thus, for real-time navigation, the simple and computationally efficient *Reactive Paradigm* is often adopted. This paradigm dispenses with the planning step altogether and directly associates an action with a perception (Figure 1.1b). Remaining insensitive to the global arrangements of objects and considering only the

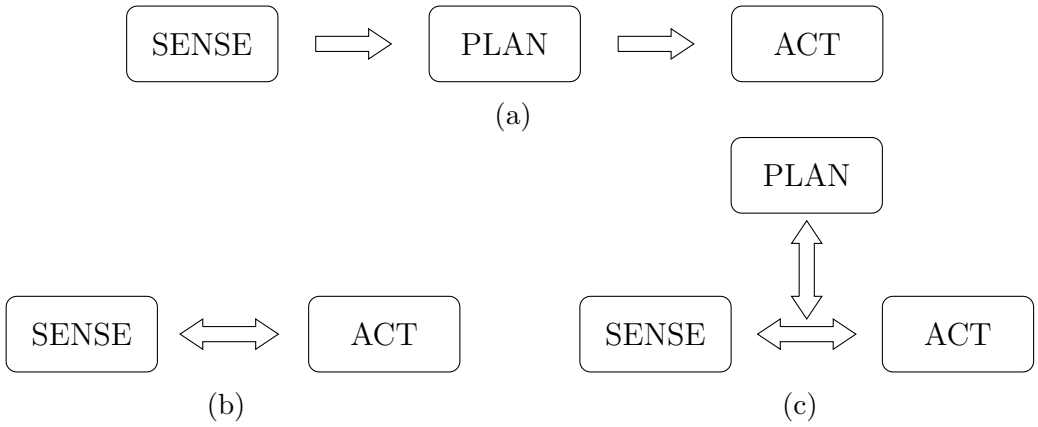


Figure 1.1: Depending on the way the modules of sense, plan and act are organized, different paradigms in navigation arise (a) Hierarchical Paradigm; (b) Reactive Paradigm; and (c) Hybrid Deliberative/Reactive Paradigm

goal position and obstacles in the immediate vicinity, it prescribes a reactive motion to a perception. Lacking in global perspective and memory of its past actions, the robot in this approach is susceptible to getting trapped into continued oscillations or cycles. Well known methods for reactive navigation include [57, 58]. A neural network, trained with the back-propagation algorithm, is well-suited for reactive navigation as it can directly output the desired robot motion based on the goal position and the input sensor readings [59]. The limitation of a purely reactive navigation is overcome in the *Hybrid Deliberative/Reactive Paradigm*, in which the robot reacts to its sensory input (as in the Reactive Paradigm) but additionally also executes motions prescribed by a higher level deliberative planner (Figure 1.1c). The planner receives its input from the same set of sensors as the action module and is responsible for higher-level tasks like map-building, localization and path-planning. Thus, the robot remains committed to its long-term plan but is also quick enough to react to sudden changes in its vicinity [60]. The planner is typically invoked after every 300–500 cycles (exact interval of invocation depends on the speed of the robot and environment) of the sense–act step and thereby ensures that navigation is not significantly slowed down by the compute-intensive path-planning phase.

While the robot pursues the prescribed path, it usually strays from it because of imperfections and noise in actuation. So, it is necessary to correct its course of travel periodically to bring it back to the desired path. This is called *trajectory tracking* and a simple method to achieve this is through the use of the *Pure-pursuit algorithm* [61]. Using an estimate of its current position, this algorithm computes the curvature of the path along which the robot should move to arrive at the desired path at a given look ahead distance.

While the robot moves in its environment, it is necessary that it continually esti-

mates its position and heading direction (or orientation), collectively termed as *pose*. Information on its pose allows the robot to determine whether it has reached its goal, is on the prescribed path or has deviated from it and the like. This capability of the robot to continually estimate its pose is called *localization* and is a key requirement for autonomous navigation. It is usually not possible to accurately measure the pose of a robot directly; it can only be inferred or estimated. Though, *Global Positioning Systems* (GPSs) do provide reasonable estimates of robot position, they are not usable in indoor or in GPS-denied environments. However, if a good model of its environment is available with robot in the form of map, and the robot is able to sense its distance from the objects in its immediate vicinity, it can make an estimate of its pose. In order that the map faithfully models the environment, it is imperative that the spatial locations of the objects are correctly known. This is possible only if the robot knows its pose from where it perceived the objects during the process of map building. But knowing the pose requires a map of the environment. This, in essence, implies that map building and localization are inter-dependent processes. For localization, a map must be available; conversely, for building a map, the robot must be well localized. Thus, we are faced with a chicken-and-egg problem, to solve which the problems of mapping and localization are generally adressed concurrently.

Figure 1.2 illustrates the key steps in autonomous navigation. A robot utilizes a sensor like LRF to perceive its environment for building a map and localizing itself with respect to the map. (It is important to note that localization and map building are just two instances of robot perception; depending upon the task at hand, the outcome of perception may differ.) The robot can also utilize the same map, or some suitably modified form of it, to plan a path from its current state to its goal state. The trajectory tracking mechanism ensures that the robot stays on the planned path. The motion control system accordingly actuates the locomotion mechanism to take the robot towards its goal.

1.4 Map building and localization

A mobile robot, if it is to be deployed for any meaningful purpose like radiation monitoring or material transfer, must have the capability of navigating autonomously in its environment. One of the necessary prerequisites for autonomous navigation is the capability of localization [62], which continually provides the robot with an answer to the question, *Where am I?* In other words, it enables the robot to reliably estimate its location and orientation (heading direction) in its environment with respect to a world coordinate system.

If the pose (location and orientation) of a robot is known at any instant, the subsequent poses may be estimated through odometry. But, this estimate soon becomes

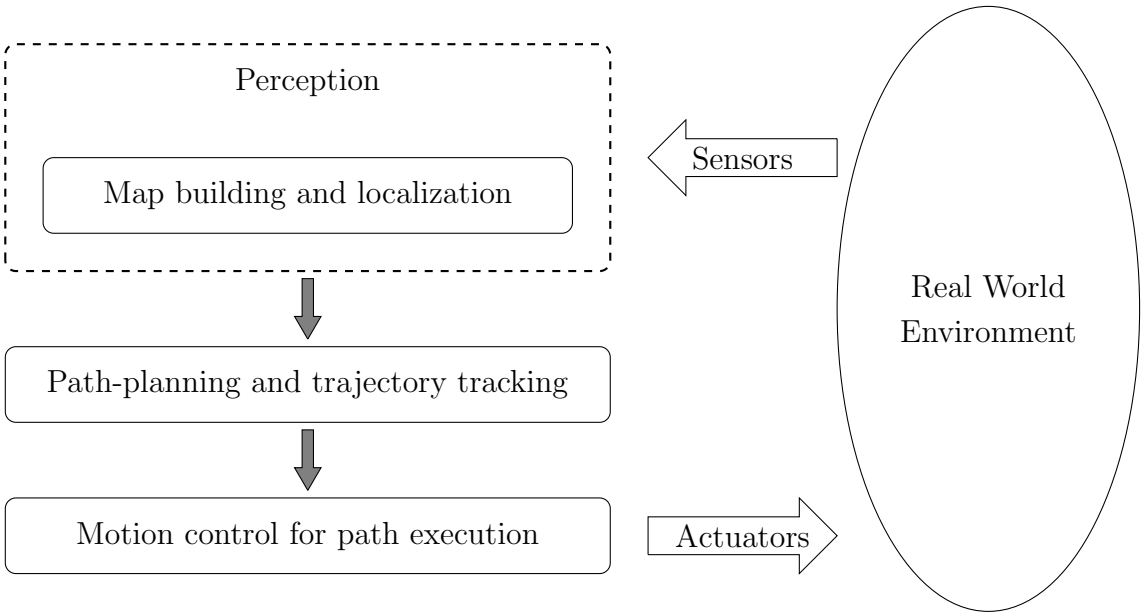


Figure 1.2: Key steps in autonomous navigation

unreliable because of errors that accumulate over the length of travel. Nonetheless, if the spatial locations of the objects in the vicinity of the robot are known, and the robot is able to measure its distance from these objects using sensors like LRF, it is possible to make an estimate of its pose. Thus arises the need of having a map of the environment of the robot [63]. A map is essentially a *spatial model*, or an abstraction of the environment, where (almost) all pertinent objects in the environment find a representation in one form or the other, depending upon the type of the map. A map is a *representation* of the environment that is used internally by the robot. The form of representation used in a map is crucial as it has a bearing on how efficiently it can be stored and how conveniently it can be put to the desired use. The sensor used to build a map often guides the choice of a particular map representation.

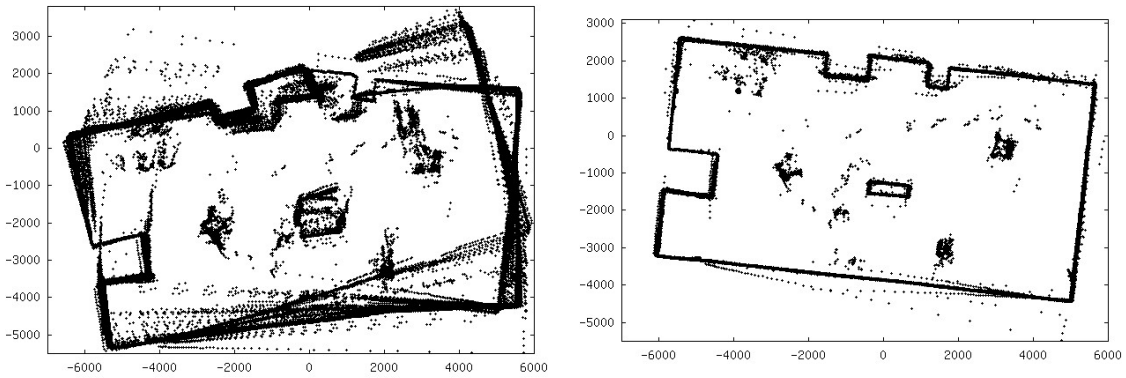
Two different paradigms for indoor maps are common: *metric maps* and *topological maps*. In a metric map (also called *geometric map*), locations of objects in the physical environment are described in a world coordinate system, though the representations of objects differ according to the type of map. In such maps, the distance between any two points in the map corresponds to the distance between the corresponding two physical points in the real world. This makes it possible for a mobile robot to figure out its pose by consulting the map of the environment after it has sensed its distance from the neighboring obstacles. Such maps are preferable when precise localization and accurate path-planning are desired. Metric maps are of two types: *grid maps* and *feature maps*. In a grid map, the environment is described by a two-dimensional grid, where each cell of the grid is either filled (indicating presence of an object in the corresponding part of the environment) or empty (indicating free space). Such a grid map is commonly

referred to as *occupancy grid map* [64, 65]. In place of occupancy, values indicating the amount the corresponding cell is covered by an obstacle is considered in [66]. A major drawback of the grid-based representation is that a large amount of memory is necessary to store the map. A feature map is a collection of features, where each feature is an abstraction of a distinct entity of the mapped environment and has an associated spatial location. Features are often described by geometric primitives like line-segments and arcs or by more abstract representations like doors and tables. Such abstract descriptions permit compact and scalable representation of the environment compared to occupancy grid representations. Topological maps [67], capture the structure and connectivity of the environment through the use of graphs. Nodes of the graph represent distinct places, while arcs represent spatial relationships between places. The advantage of the topological maps lies in their compactness and amenability to efficient planning. To gain the best of both these paradigms, metric and topological maps are combined in [68].

LRFs have become the sensor of choice for range finding in mobile robotics. These sensors provide dense and accurate range data at a very high resolution and sampling rate. A laser range scan may be thought of as giving a partial view of the environment. By integrating many such views taken from different but known locations, a model of the complete environment (which is commonly referred to as a *global map*) can be obtained. Clearly, this entails that poses from which the scans are taken (called *scan poses*) are accurately known. But this is possible only if the robot can localize itself accurately, which again calls for the availability of an accurate map. In the absence of both a map and an exact pose information, the robot is required to simultaneously estimate both the map and localize itself relative to the (possibly partial) map built thus far—a problem known in the robotics community as the problem of *Simultaneous Localization and Mapping* (SLAM) [69, 70].

For the purpose of map building, a robot collects laser range scans while exploring its environment. A raw scan is usually filtered and processed for extraction of higher-level features like line segments. In the *incremental* scheme of map building (typically done *online*), the scans (or the features extracted from it, as the case may be) are successively merged with a cumulatively built global model of the environment. Thus, more and more information about the environment, acquired by the robot in steps, are incrementally added to the map to get a progressively more complete description of the environment. In an incrementally built map, the range scans/features acquired from different places in the environment are often integrated into a common world coordinate frame using odometry-estimates of the scan poses. Since these estimates are corrupted by cumulative drift errors, the spatial relationships between the scan poses estimated through odometry are inconsistent. Hence, the range scans/features integrated using odometry estimates of the scan poses are grossly misaligned. Consequently, the result-

ing map does not give a faithful representation of the actual environment (Figure 1.3a). It is, thus, essential to register the scans/features (by obtaining improved estimates of the scan poses) prior to their integration for building the map. A map built from pre-registered scans models the environment rather closely (Figure 1.3b). (We shall revisit this environment once again in Section 3.2.2 (page 40) for map building). Thus, consistent registration of scans is a crucial issue in the process of map building. The method proposed by Lu-Milios [71] takes in a set of odometry-estimated scan poses and the laser range scans taken from those poses to provide improved estimates of the scan poses for globally consistent range scan alignment.



(a) (b)

Figure 1.3: An incrementally built point cloud-based map. (a) The range scans have been integrated using odometry-estimated scan poses; (b) The range scans have been registered before integration.

If a map of the environment is available *a priori*, the task of localization becomes comparatively straightforward. In that case, a scan (or features extracted from it) obtained from an unknown pose may be rotated and translated so that it matches with the map. The amount of translation and rotation required to achieve this matching gives an estimate of the difference in position and orientation of the mobile robot from the last known position and orientation. Using this approach, it is possible to continuously track the pose of the robot and achieve what is commonly known as *pose-tracking* or *local localization*. But it assumes that the initial position of the robot is known. In this approach, if at any stage the estimated pose of the robot turns out to be widely different from the actual, pose-tracking may fail in a subsequent step and the robot may get lost. In instances, when the robot is not aware of its initial position except that it is located somewhere on the map, the robot infers its pose by matching the obtained scan with different regions of the map and identifying the object or region with which the closest match is obtained. Pose estimation in such cases is known as *global localization*. Global localization is, in general, difficult because different regions of the same environment often look identical locally, a phenomenon generally known as

perceptual aliasing. Solution to the global localization problem requires that multiple hypotheses about the pose of the robot is maintained and the *beliefs* corresponding to the hypotheses that they actually represent the true robot pose are updated with the arrival of new information. Such problems are best solved in a probabilistic framework in which the robot pose is modeled as a random variable with an associated probability density. The problem of global localization degenerates to one of tracking, once the robot is localized. A related problem is that of the *robot kidnapped problem*, in which the robot believes that it knows its pose, while in reality, it does not.

1.5 Motivation, scope and organization of the thesis

In many real-life applications, a robot is required to navigate continuously in environments that remain practically invariant over long periods of time. For all these applications, it is more appropriate to build a map of the working environment first and then use it for localization, rather than operate the robot in an online SLAM mode, which is complex, computation-intensive and less accurate. By decoupling mapping from localization, the robot can be absolved of the job of map building while it is performing its assigned duty.

Applications of the above kind require that a method is available for offline building of maps from range data collected by the robot during an initial phase when it is manually driven or tele-operated in its environment. A robust and accurate method for localization on maps built by such a method is also a prerequisite for the robot to move around safely and efficiently in the environment for performing its intended task.

With the objective of achieving robust and accurate localization of mobile robots equipped with LRFs in indoor environments that remain practically unchanged for extended periods of time, in this thesis we propose

- an offline method for building maps of indoor environments by merging line segments extracted from registered laser range scans [72]; and
- a robust and accurate method for pose-tracking on such maps based on the Monte Carlo framework [73].

Line segments are an obvious choice in the representation of indoor environments, which predominantly comprise objects like walls, corridors and cupboards. These objects have planar external surfaces and thus naturally lend themselves to representation by line segments in the 2D-plane. Moreover, maps based on line segments are compact, provide floating-point resolution and scale well with the environment size. In simple point landmarks-based maps, the landmarks are often not visible due to occlusion and

are also difficult to uniquely identify as they lack distinguishing features like orientation and length. Hence, in this thesis, we limit our scope to building line segment-based maps of indoor environments only. We experimentally compare the maps produced by our method with those produced by two other methods [74, 75] reported in the literature. We also propose simple ways of quantitatively assessing the goodness of a line segment-based map produced in relation to the ground truth.

Monte Carlo Localization (MCL) is a popular and powerful approach for mobile robot localization [76]. It uses a finite set of random samples with associated *importance weights* to represent, in approximation, the posterior probability density function of the robot pose, over the state space. The benefits associated with the use of line segment-based maps and the effectiveness of MCL [77] are sufficient motivations for implementing MCL on such maps. But MCL has seldom been studied in the context of line segment-based maps. A key step of the approach—and one that can endow it with or rob it of the attributes of accuracy, robustness and efficiency—is the computation of the so called importance weights associated with each hypothesized pose. In this thesis, we propose a formulation for MCL that uses a novel method for the computation of importance weights on maps represented with line segments, and extensively study its performance in pose-tracking. The proposed method does not require modification of the environment, through the placement of reflectors or guidewires, for instance, to simplify the task of localization. We also compare our method with three other competing methods [78–80] in the literature and present the results and insights thus gathered.

The remainder of this thesis is organized as follows: in Chapter 2 we review literature relating to different aspects of building maps from line segments, including methods for extraction of line segments from laser range data, curve approximations, line segment-based map building and line segment merging. Thereafter, we describe the proposed method of map building in details after giving a brief overview of mean-shift clustering.

Chapter 3 presents the details of the assessment and comparative study of the maps produced by the proposed method and by two other methods [74,75] from the literature. Two approaches for assessment of line segment-based maps are also presented in this Chapter.

In Chapter 4 we review literature on different approaches to localization and then discuss on the framework of particle filtering on which MCL is based. We then present the details of the proposed method of weight computation for MCL and briefly describe the weight computation procedure of three competing methods [78–80] from the literature.

In Chapter 5 we present the details of the procedures adopted to evaluate, and the results of evaluation, of the performance of MCL incorporating the method of weight

computation proposed in Chapter 4 as well as those incorporating the methods reported in [78–80].

Chapter 6 summarizes the contribution of the thesis, the conclusions arrived at and the possible themes for future research.

Chapter 2

Building maps using line segments

2.1 Introduction

We have seen in the preceding Chapter that a key capability for autonomous navigation of a mobile robot is that it is continuously able to localize itself in its environment. We have also seen that a map of the environment should be available with the robot so that it is able to localize itself by matching its perception of the environment with the map. Since the process of map building and that of localization are closely intertwined, they are commonly solved using SLAM methods [69, 70].

In many real-life situations, the robot is required to navigate continuously in environments that remain practically invariant over long periods of time. One instance of such situation is the requirement of 2D profiling of radiation inside the vault of a cyclotron during its operation using a radiation monitor piggybacked on a mobile robot [81]. Another instance could be the continuous transportation of materials by automated guided vehicles in a factory floor [35]. In all these cases, it is more appropriate to build a map of the working environment first and then use it for localization, rather than operate the robot in an online SLAM mode, which is complex, computation-intensive and less accurate. To localize itself, the robot needs to search for a plausible pose—a pose that best satisfies the sensor readings of the robot—in the space of all possible poses. To build the map, it needs to ascertain whether or not it is in a part of the environment visited previously and then update the map accordingly. If these two tasks are decoupled, and the complete map is built independently in an initial offline phase, the robot can be absolved of the computational burden of map building during its operational phase. With this motivation in the backdrop, in this thesis, we propose an *offline* method for building global maps of indoor environments using directed line segments extracted from laser range data. The method is offline in the sense that processing commences after all data are collected. This is in contrast to *online* methods (typically SLAM) that process sensor data as and when they arrive and use them to

incrementally update the map.

Line segments are an obvious choice in the representation of indoor features. Most indoor environments predominantly comprise objects like walls, corridors and cupboards. These objects have planar external surfaces and thus naturally lend themselves to representation by line segments in the 2D-plane. Unlike occupancy grids, maps based on line segments are more compact, provide floating-point resolution, consume significantly less memory and thus scale well with the environment size. Formally, we define a line segment-based 2D map M as a set of N line segments such that $M = \{L_i\}_{i=1}^{N}$ where each segment L_i has a specific orientation, defined by its start point and end point. In this Chapter, our focus shall be on building line segment-based maps of indoor environments in the 2D plane using laser range data.

Methods for building line segment-based maps typically include the following three steps: (a) registration of range scan data; (b) extraction of line segments from the range data; and (c) selective merging of the line segments so extracted. Even though, we propose a new formulation for step (c) only and use pre-existing algorithms for the first two steps, nonetheless, our method coherently combines all the three steps to generate compact and accurate line segment-based maps starting from raw laser range data.

In the proposed method, the robot is first manually driven (or tele-operated) through the entire environment envisaged to be mapped. As the robot is driven, laser scan data and the corresponding scan poses obtained through odometry are recorded at regular intervals. The scan poses are then adjusted and optimized for consistent registration of range scan data using the Lu-Milios algorithm [71] (see Section 3.2 (page 32) for a more detailed discussion). Once the corrected scan poses are obtained, line segments are extracted from the laser range data acquired from each of the poses. The entire gamut of the line segments is then subjected to two successive steps of density-based clustering. These two steps help delineate the line segments that are in close proximity to each other. All line segments that are in close proximity, and hence represent the same object, are finally merged together to yield a (resultant) line segment of the map.

2.2 Related work

In this Section, we glance through some work on the extraction of line segments from laser range data and the related domain of approximation of digitized curves. Thereafter, we review a few methods for line segment-based map building, focusing particularly on the aspects of scan registration and line segment merging.

The domains of pattern recognition and computer vision have contributed most of the early methods for extraction of line segments from laser range data. The method of Iterative-End-Point-Fit [82] and its variant, the Split-and-Merge algorithm [83], re-

cursively splits a line segment fitted to a set of points till the distance of the farthest point from the fitted line falls below a threshold. Consecutive line segments are merged if they are nearly collinear. Nguyen et al. [84] experimentally evaluated six different classes of line extractions algorithms viz., the Split-and-Merge Algorithm [83, 85], the Incremental Algorithm [86], the Hough Transform Algorithm [87], the Line Regression Algorithm [88], the RANSAC (Random Sample Consensus) Algorithm [89] and the EM (Expectation Maximization) Algorithm [90] as applied to mobile robotics using 2D laser range data acquired in indoor environments. The results of these algorithms were compared with the ground truth to assess their efficacy. They concluded that Split-and-Merge is superior to the other algorithms in terms of speed and correctness. In more recent years, Harati and Siegwart [91] proposed a simple mechanism for line extraction based on the thresholding of *Bearing Angle*, which is the angle between the laser beam and the line passing through consecutive scan points. They also proposed a framework for hierarchic representation of a 2D range scan at several levels-of-detail. Fernández et al. [92] proposed two methods that are to be applied on laser range data in succession—the first, Distance-based Convolution Clustering, to cluster the scan data points; and the second, Reduced Hough Transform Line Tracker, to fit a line to each cluster. The method proposed in [93] predicts the position of a scan point based on the preceding points and uses it to detect the presence of discontinuities and turns in the environment. Subsequently, it fits line segments and circles/arcs to the scan data as appropriate to model the environment. Line fitting to a set of points is commonly carried out using total-least-squares fitting [88, 94].

Closely related to the problem of extraction of line segments from laser range data is the problem of representing digitized curves through line segments and other higher-order geometric primitives like circular arcs and splines. Unlike laser range data, the points representing a digitized curve are always 8-connected and the end points of the geometric primitives belong to the set of points defining the curve. When a closed curve is approximated by a set of line segments, the problem is commonly referred to as that of *polygonal approximation*. Over the years, many algorithms have been proposed for polygonal approximation. The sequential methods for polygonal approximation continuously evaluate a given approximation metric as they perform a linear scan along the length of the curve to locate an approximation line segment e.g. [95–97]. The dominant point detection-based methods compute the curvature of each point of the curve and then designate the local curvature-maximum point within a given neighborhood as the dominant point. The polygonal approximation is obtained by joining the dominant points [98–101]. Methods for polygonal approximation based on *metaheuristics* [102] have also been proposed. These include methods based on Genetic Algorithm [103–105], Tabu Search [106], Ant Colony Optimization [107], Particle Swarm Optimization [108], Differential Evolution [109] and Artificial Bee Colony Algorithm [110]. Methods for

approximation using line segments and circular arcs include [111–113]. Curves have also been represented using a *hierarchical* approach as in [114].

A plethora of methods, either online or offline, for map building from line segments have been proposed over the years [115]. The online or incremental methods either do not include any scan registration step [116] or register the scans as and when they arrive with the so far accumulated scans [87, 117, 118]. For instance, the Consecutive Clustering Algorithm [116] successively applies clustering methods, first on raw scan data (K-means), then on the tentative line segments fitted to this data (K-means) and subsequently for finding similar line segments for merging them to the map (Rank Order Clustering). Relying on the assumption that no localization error accrues during collection of the range data, the authors did not address the issue of scan registration. In [119], a set of line segments extracted from the recent scan is registered with the partial line segment-based map built so far, before subsuming the line segments into the map. The offline methods, on the other hand, register all scans simultaneously so that the scans are globally consistent. The method presented in [120] registers line segments extracted from laser scan data, after elimination of redundant scans. Using a measure of similarity between line segments, they are clustered and then merged. In the final step, inconsistent line segments are eliminated using the notion of *sight triangles*. A method for building global maps by integrating partial maps consisting of line segments only, without taking recourse to pose information, is presented in [121].

In the more recent years, methods have been proposed that identify and merge redundant line segments in maps with the aim that each planar surface is represented by a single line segment. A comparison of some such methods is reported in [122]. They categorize the existing methods of map segment merging into one of three classes, where: (i) consecutive/collinear line segments are merged; (ii) pairs of line segments are merged; and (iii) sets of line segments are merged. They conclude that unlike methods of the last class, redundant line segments are left in maps that employ merging methods of the first two classes. The method proposed in [74] first identifies the longest line segment and then defines a strip of width 2ε (where ε is a user-settable parameter of the method) centered on the segment. All line segments whose end points lie within the strip and whose projection on the longest segment form a continuous *line segment chain* are merged to yield the resultant single segment. In the algorithm proposed in [75], segments are clustered, using mean-shift clustering with a symmetric Gaussian kernel, to segregate all spatially close segments that share a more-or-less common direction. Segments from such clusters are further clustered, using mean-shift clustering but this time with an anisotropic Gaussian kernel, to identify segments that are collinear or roughly so. The resulting segments are then merged to yield the final line segment.

2.3 Mean-shift clustering: An overview

Mean-shift clustering [123–126] is a density-based clustering technique. The modes of the probability density function underlying the distribution of the given data points are considered as the cluster centers. These modes are the points of local maxima of the density function. The mean-shift clustering technique neither requires the number of clusters to be known *a priori* nor does it impose any constraint on the shape of the clusters.

Given N instances (or samples) $\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_N, \in S = \mathfrak{R}^d \quad (d > 0)$, of the random variable $\boldsymbol{x}$, the probability density function, $\widehat{f}_k(\boldsymbol{x})$, can be estimated using the Parzen window technique [82] with a kernel function $K : S \rightarrow \mathfrak{R}$ defined as

$$K(\boldsymbol{x}) = c_{k,d}\, k(\| \boldsymbol{x} \|^2) \geq 0 \tag{2.1}$$

where $k : [0, \infty] \rightarrow \mathfrak{R}$ is the kernel profile and $c_{k,d}$ is a normalization constant that ensure

$$\int_{\mathfrak{R}^d} K(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = 1 \tag{2.2}$$

The kernel density estimate is then given as:

$$\widehat{f}_K(\boldsymbol{x}) = \frac{c_{k,d}}{N \cdot h^d} \sum_{j=1}^{N} k\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j) \tag{2.3}$$

where $h > 0$ is the *window width*, also called the *bandwidth*, and $w(\boldsymbol{s}_j)$ is the weight associated with the sample point $\boldsymbol{s}_j$. Assuming that the kernel profile $k(\boldsymbol{x})$ is differentiable, the gradient of the density function is given by

$$\begin{aligned} \nabla \widehat{f}_K(\boldsymbol{x}) &= \frac{2\, c_{k,d}}{N \cdot h^{d+2}} \left[\sum_{j=1}^{N} (\boldsymbol{x} - \boldsymbol{s}_j) \cdot k'\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j) \right] \\ &= \frac{2\, c_{k,d}}{N \cdot h^{d+2}} \left[\sum_{j=1}^{N} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j) \right] \times \\ &\qquad \left[\frac{\sum_{j=1}^{N} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j) \cdot \boldsymbol{s}_j}{\sum_{j=1}^{N} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j)} - \boldsymbol{x} \right] \end{aligned} \tag{2.4}$$

where

$$g(\cdot) = -k'(\cdot) \tag{2.5}$$

With analogy from equation (2.3), equation (2.4) may be re-written as

$$\nabla \widehat{f}_K(\boldsymbol{x}) = 2\, \widehat{f}_G(\boldsymbol{x}) \cdot \left[\frac{\sum_{j=1}^{N} g\left(\left\| \frac{\boldsymbol{x}-\boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j) \cdot \boldsymbol{s}_j}{\sum_{j=1}^{N} g\left(\left\| \frac{\boldsymbol{x}-\boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j)} - \boldsymbol{x} \right] \tag{2.6}$$

where $\widehat{f}_G(\boldsymbol{x})$ is the probability density function estimated with the kernel function $G : S \to \Re$ defined as

$$G(\boldsymbol{x}) = c_{g,d}\; g(\| \boldsymbol{x} \|^2) \tag{2.7}$$

where $g(\cdot)$ is the kernel profile defined in equation (2.5) and $c_{g,d}$ is the normalization constant that ensures that G integrates to one.

A closer look at equation (2.6) reveals that the quantity inside the square brackets is a vector—called the *mean-shift vector*—that points in the direction of the gradient of $\widehat{f}_K(\boldsymbol{x})$ at $\boldsymbol{x}$. The magnitude of the vector is proportional to the ratio of the gradient of $\widehat{f}_K(\boldsymbol{x})$ to the density estimated at $\boldsymbol{x}$ using kernel G. The mean-shift vector at $\boldsymbol{x}$ is given as

$$m_k(\boldsymbol{x}) = \frac{\sum_{j=1}^{N} g\left(\left\| \frac{\boldsymbol{x}-\boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j) \cdot \boldsymbol{s}_j}{\sum_{j=1}^{N} g\left(\left\| \frac{\boldsymbol{x}-\boldsymbol{s}_j}{h} \right\|^2 \right) \cdot w(\boldsymbol{s}_j)} - \boldsymbol{x} \tag{2.8}$$

Thus, for some value $\boldsymbol{x}_i$ of $\boldsymbol{x}$, if we iterate through the following steps (t is the iteration counter):

1. Compute the mean-shift vector $m_k(\boldsymbol{x}_i{}^{(t)})$.
2. Shift the sample point $\boldsymbol{x}_i{}^{(t)}$ by an amount $m_k(\boldsymbol{x}_i{}^{(t)})$ so as to reach

   $$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + m_k(\boldsymbol{x}_i{}^{(t)}).$$

3. Increment t and go to (1) until $\boldsymbol{x}_i^{(t+1)} \approx \boldsymbol{x}_i^{(t)}$.

$\boldsymbol{x}_i$ eventually reaches a mode of the density function. The condition $\boldsymbol{x}_i^{(t+1)} \approx \boldsymbol{x}_i^{(t)}$ in step (3) indicates that the algorithm has converged [125] to a mode of the density function, since at the modes $m_k(\boldsymbol{x}_i) = 0$. Thus, the mean-shift procedure is a hill-climbing technique which takes $\boldsymbol{x}_i$ to the nearest local maximum (or mode) of the density surface in whose *basin of attraction* $\boldsymbol{x}_i$ lies. Thus, when the above iterative steps are performed on all the sample points one-after-another (or in parallel), they all converge to one mode or the other depending upon their basin of attraction. All sample points converging to the same mode belong to the same cluster.

Weights associated with sample points influence the shape of the estimated probability density function. The greater is the weight, $w(\boldsymbol{s})$, associated with a sample point, $\boldsymbol{s}$, the greater is the "pull" exerted by the sample point in bringing the local maxima, in whose basin of attraction it lies, towards itself. In other words, it tries to bring the centre of the cluster to which it belongs towards itself.

In the so-called *blurring mean-shift*, in iteration t, a sample value $\boldsymbol{s}_i^{(t)}$ is substituted for $\boldsymbol{x}$ in equation (2.8). The sample value then gets updated as: $\boldsymbol{s}_i^{(t+1)} = \boldsymbol{s}_i^{(t)} + m_k(\boldsymbol{s}_i{}^{(t)})$. This is done separately for all N-instances of the sample points i.e., $i = 1, 2, \ldots, N$ till convergence. In this approach, the density function $\widehat{f}_K(\boldsymbol{x})$ changes in each iteration as all $\boldsymbol{s}_j$ s in equation (2.3) change.

In the *non-blurring mean-shift*, a sample value $\boldsymbol{s}_i$ is assigned to $\boldsymbol{x}$ only in the initialization step as follows: $\boldsymbol{x}_i^{(0)} \leftarrow \boldsymbol{s}_i,\ i = 1, 2, \ldots, N$. In subsequent iterations, the point $\boldsymbol{x}_i$ is updated as: $\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + m_k(\boldsymbol{x}_i{}^{(t)}),\ i = 1, 2, \ldots, N$ till convergence. The sample points themselves hold on to their values. In this approach, the density function $\widehat{f}_K(\boldsymbol{x})$ does not change from iteration to iteration.

It is important to note that the first term in the right-hand-side of equation (2.8), and hence, the expression for the mean-shift vector being independent of the normalization constant, for all practical purposes we work with the simplest possible expression for the kernel function by relaxing the requirement that it integrates to one. In the following, we list some of the commonly used kernel functions, in their simplest form, having unit bandwidth.

$$\text{Flat Kernel:} \qquad F(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \|\boldsymbol{x}\| \leq 1, \\ 0 & \text{if } \|\boldsymbol{x}\| > 1. \end{cases} \tag{2.9}$$

$$\text{Gaussian Kernel:} \qquad G(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2} \tag{2.10}$$

$$\text{Epanechnikov Kernel:} \qquad E(\boldsymbol{x}) = \begin{cases} 1 - \|\boldsymbol{x}\|^2 & \text{if } \|\boldsymbol{x}\| \leq 1, \\ 0 & \text{if } \|\boldsymbol{x}\| > 1. \end{cases} \tag{2.11}$$

$$\text{Bi-weight Kernel:} \qquad B(\boldsymbol{x}) = \begin{cases} (1 - \|\boldsymbol{x}\|^2)^2 & \text{if } \|\boldsymbol{x}\| \leq 1, \\ 0 & \text{if } \|\boldsymbol{x}\| > 1. \end{cases} \tag{2.12}$$

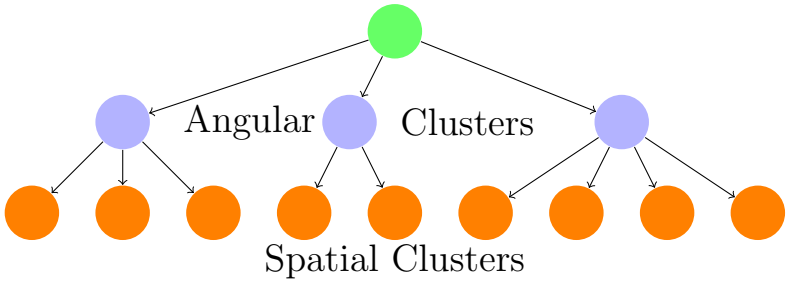


Figure 2.1: Organization of the scan line segments in a tree-like hierarchy.

2.4 The proposed method

The proposed method starts off by extracting line segments from all individual laser range scans. Each such line segment—which we call *scan line segment* (or *scan segment*, in short)—may be thought of as a *representative* of that part of the *linear feature* of the environment that is *visible* from the corresponding scan pose. We use the term linear feature to refer to a planar (or a piecewise planar) surface of the environment that is opaque to laser e.g., the front face of a closed rectangular cupboard or a continuous corridor wall. Since a given linear feature is likely to be visible (either in whole or in part) from multiple scan poses, it is likely to have multiple scan segments as its representatives. Hence, the problem of line segment-based map building boils down to one of identifying all the scan segments that represent a given linear feature of the environment, and eliciting the geometric properties of the feature from the identified scan segments by merging them. A line segment-based map results when the geometric properties of all the relevant linear features of the environment are identified from the corresponding set of scan segments.

The proposed method organizes the scan segments in a tree-like hierarchy (Figure 2.1). At the top of the hierarchy is a node that represents all scan segments extracted from all the range scans. In the next level, these scan segments are distributed over one or more nodes, where each node holds closely oriented scan segments. For convenience, we refer to all scan segments in a node at this level as belonging to an *angular cluster*. At the next lower level, scan segments of each angular cluster are distributed over one or more leaf nodes, where each leaf node holds scan segments that lie close to each other in space. All scan line segments of a given leaf node are considered as representatives of a distinct linear feature and are said to belong to the same *spatial cluster*. The geometrical attributes (length, orientation and position) of the linear feature represented by a spatial cluster may be obtained by suitably merging the scan lines segments belonging to the cluster into one segment.

```
1: extractLineSegments (scanPoints: r[ ], segmentIndex: start, segmentIndex: end)
2: if NOT( ( D(r[start], r[end]) >=100 ) AND (end - start >= 5 ) ) then
3:     return (NULL, NULL)
4: else
5:     maxIndex = index ∈ {start+1, end-1} | D⊥(r[ ], index, start, end)>= D⊥(r[ ],
   i, start, end) ∀i ∈ [start+1, end-1]
6:     maxDist = D⊥(r[ ], maxIndex, start, end)
7:     if (maxDist > 50) then
8:         extractLineSegments (r[ ], start, maxIndex)
9:         extractLineSegments (r[ ], maxIndex, end)
10:    else
11:        L=total-least-square-fitting (r[ ], start, end)
12:        segStart = project (r[start], L)
13:        segEnd = project (r[end], L)
14:        return (segStart, segEnd)
15:    end if
16: end if
```

Figure 2.2: Pseudo-code for extraction of scan line segments from a given laser range scan.

2.4.1 Extraction of line segments from individual scans

We assume that while the robot moves in its environment, it keeps collecting laser range scan data in the counterclockwise sense, covering an angular range of 180°, centered on its heading direction. In each scan, whenever we come across a range value that is greater than ρ_{max} $(= 8000mm)$ or is less than ρ_{min} $(= 50mm)$, we discard the corresponding data from further processing. If we suppose that n number of range points remain in a scan after discarding, the scan may be defined as a set of n points given by $\{r_i \mid r_i = (\rho_i \cdot cos\ \psi_i,\ \rho_i \cdot sin\ \psi_i)\}_{i=1}^{n}$ where ρ_i $(\rho_{min} \leq \rho_i \leq \rho_{max})$ is the range value acquired at an angle ψ_i during scanning. Before extraction of line segments commences, starting from $i = j$ $(1 < j < n)$, the distance between two consecutive scan points r_{i-1} and r_i, denoted by $\delta_{i-1,i}$, is computed successively till $\delta_{i-1,i} > 100$mm for some $i = k$ where $j \leq k \leq n$ or till $i = n$. The indices $i = j - 1$ and $i = k - 1$ (or $i = j - 1$ and $i = n$, if the second condition is true) define a closely-spaced ordered subset of scan points within which line extraction is to be attempted. Search for the next subset of closely-spaced points starts with $j = k + 1(k < n)$, and so on.

In order to extract the so called scan line segments from a closely-spaced ordered subset of scans points, we broadly follow the pseudo-code, derived from the Iterative End Point Fitting algorithm [82] with worst case complexity $O(n^2)$, given in Figure 2.2

The function `extractLineSegments( )` takes in as arguments, an ordered set of range points `r[ ]` and the indices, `start` and `end`, of two points in the set, with `start < end`. These indices define an ordered subset of closely-spaced scan points

on the set `r[ ]` within which line segment extraction is attempted. The function `D( )` computes the Euclidean distance between the two range points passed in as arguments to it. If this distance is less than 100mm or the number of range points in the subset under consideration is less than 5, we do not attempt extraction of line segments from the corresponding subset. The entire subset is discarded as segments extracted from it would be too short. Otherwise, we successively compute the distances—using the function $\texttt{D}_{\perp}\texttt{( )}$—of all the intermediate scan points from the line joining the points with indices `start` and `end`, and note the point with the maximum absolute distance. If this distance is greater than 50mm, line segments are extracted recursively from the two subsets of scan points demarcated by the point with the maximum absolute deviation. If the distance is not greater than 50mm, total-least-squares fitting [88, 94] is applied on the corresponding subset of scan points to get the line `L`. Thereafter, the scan points with indices `start` and `end` are projected on `L`, to determine the two end points of `L` and to convert it into a line segment. The end points that define the line segment are finally returned.

The scan line segments extracted from the individual scans, which are shorter than 100mm are discarded. The end points of the remaining segments are then transformed to a (common) world coordinate frame. The transformation is carried out using the scan pose of range scan data from which the segment in question has been extracted. The scan poses are estimated through odometry and optimized by scan registration algorithm. The entire gamut of all such extracted scan line segments is held in the root node of Figure 2.1.

2.4.2 Orientation-based clustering of scan line segments

In trying to classify the scan segments that represent one or the other linear feature of the environment, we first compute the angular orientations of the scan segments. The orientation of each segment, denoted by θ, is computed in the range $-\pi < \theta \leq \pi$. It is necessary to compute the orientations in the range 2π (rather than in the range π) to preserve the sense in which the line segments were scanned by the LRF. As we shall see later, this is a vital piece of information that we can ill-afford to discard.

All scan segments arising from the same linear feature should have identical orientations. However, in practice, because of errors in the acquisition of scan points as well as in the approximation of scan points by scan line segments, the orientations show a spread or distribution, possibly skewed, around some central value. Thus, it becomes necessary to elicit the true orientations of the linear features of the environment from these distributions. We associate the true orientations of the linear features with the local maxima of the probability density function estimated using the orientations of the scan segments as samples. We take recourse to the mean-shift algorithm, discussed

in Section 2.3, not only to locate the local maxima but also to cluster the scan lines based on their orientations. We call the resultant clusters as angular clusters. All scan line segments in an angular cluster have fairly close orientations and represent linear features with similar orientations. For instance, all scan segments arising from the left and right portions of a straight corridor wall, at the center of which lies a laterally displaced door, will find themselves in the same angular cluster although they represent two different features (the left part and the right part of the wall). As a matter of fact, scan segments arising from the door will also occupy the same angular cluster, provided of course that the door is parallel to the wall. The orientation corresponding to the cluster center is a measure of the central tendency of all scan segments in the angular cluster and is regarded as the *orientation of the angular cluster*.

We invoke the mean-shift clustering algorithm on the orientation data using a bi-weight kernel. It is important to note that orientation is a circular measure and hence the distance between two orientations must be appropriately defined. We consider the distance between the angles α and θ as $1 - cos(\alpha - \theta)$. The bi-weight kernel, with bandwidth h, is defined accordingly as

$$B(\theta) = \begin{cases} \left[1 - \frac{1-cos(\alpha-\theta)}{h}\right]^2 & \text{if } \frac{1-cos(\alpha-\theta)}{h} \leq 1, \\ 0 & \text{if } \frac{1-cos(\alpha-\theta)}{h} > 1. \end{cases} \tag{2.13}$$

If $\{\theta_1, \theta_2, ..., \theta_N\}$ be the orientation data resulting from N scan line segments and $w(\theta_j)$ be the weight associated with each data point, θ_j, the underlying probability density function $\widehat{f}(\alpha)$ may be estimated as

$$\widehat{f}(\alpha) = \sum_{j=1}^{N} \left[\max\left\{\left(1 - \frac{1 - cos(\alpha - \theta_j)}{h}\right), 0\right\}\right]^2 \cdot w(\theta_j) \tag{2.14}$$

The weight $w(\theta_j)$ of the data point θ_j is set equal to the length of the scan line segment that the data point corresponds to. A data point corresponding to a scan line segment with longer length thus drives the cluster centre more towards itself.

In the non-blurring mean-shift approach, which we adopt in the proposed method, we assign each θ_j to α separately during initialization i.e., $\alpha_i^{(0)} \leftarrow \theta_i$, $i = 1, 2, \ldots, N$. Thereafter, each data point, $\alpha_i^{(t)}$, in the t-th iteration moves to $\alpha_i^{(t+1)}$according to the update rule (see [127] for further details):

$$\alpha_i^{(t+1)} = tan^{-1}\left(\frac{\sum_{j=1}^{N}\left[\max\left\{h - \left(1 - cos\left(\alpha_i^{(t)} - \theta_j\right)\right), 0\right\}\right] \cdot w(\theta_j) \cdot sin(\theta_j)}{\sum_{j=1}^{N}\left[\max\left\{h - \left(1 - cos\left(\alpha_i^{(t)} - \theta_j\right)\right), 0\right\}\right] \cdot w(\theta_j) \cdot cos(\theta_j)}\right) \tag{2.15}$$

The iterative process terminates when $\alpha_i^{(t+1)} \approx \alpha_i^{(t)}$, $i = 1, 2, ..., N$. On termination,

all the data points converge to one or the other local maximum (i.e., mode) of the probability density function. The modes correspond to the cluster centres and provide estimates of the true orientations of the predominant linear features in the environment.

So far we had tacitly assumed that when the iterations terminate, all the data points migrating towards a particular mode coalesce into the corresponding local maximum. However, in practice, after a finite number of iterations, the data points only manage to reach the vicinity of the local maximum. Of all such points in the vicinity, we reckon the one that yields the highest value of the probability density as the mode of the probability density or the cluster center.

For the sake of illustration, let us consider the histogram (Figure 2.3a) constructed with data points corresponding to the angular orientations of 8374 scan segments of a given environment. The range of the histogram is $(-\pi, \pi]$. It is evident from the histogram (in red) that the orientations of the scan line segments vary over a fairly wide range. The smooth curve (in blue) in the same Figure is the kernel density estimate of the same data computed with a bi-weight kernel with bandwidth $h = 0.02$. After 36 iterations using equation (2.15), the data points corresponding to the orientations of the scan line segments converge to one or the other mode of the density function. Figure 2.3b shows the original kernel density estimate as well as the histogram after the data points have converged. From this Figure it is clear that almost all the data points have converged to one of the four modes of the probability density function. The four modes correspond to the four predominant directions along which the linear features are oriented in the environment.

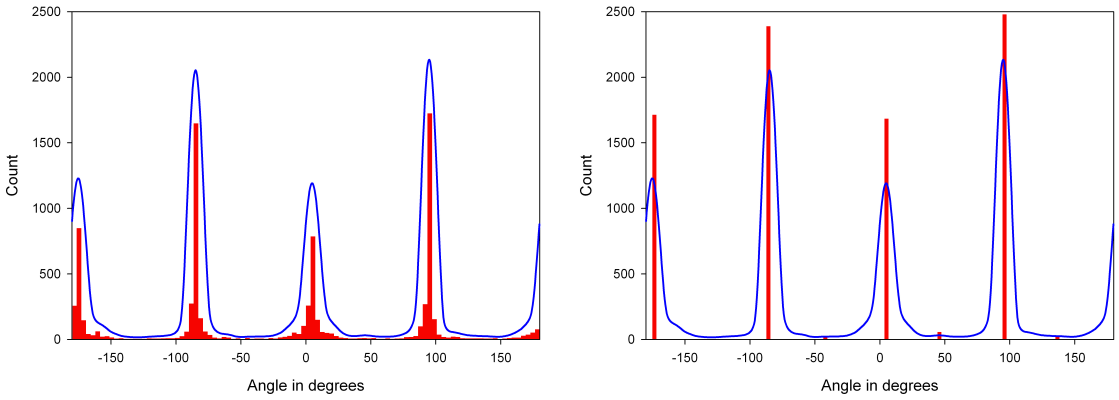
We have adopted a brute-force implementation of this step, which has a computational complexity of $O(N^2)$ where N is the total number of scan line segments.

2.4.3 Clustering of scan line segments based on spatial proximity

Scan line segments in an angular cluster are subjected to clustering so that all line segments in each resulting cluster are spatially close to one another. This step is necessary to identify scan line segments that represent different linear features but are of similar orientations. We refer to such clusters as spatial clusters. To decide if two scan line segments in an angular cluster are *spatially close*, we consider the following two measures:

1. *Lateral separation*, d: the lateral separation of two scan line segments in an angular cluster is the distance between their mid-points measured in a direction perpendicular to the orientation of the angular cluster to which they both belong.

2. *Longitudinal overlap*, p: the amount of overlap between two scan line segments



(a) (b)

Figure 2.3: The kernel density estimate (in blue), computed with a bi-weight kernel, for initial distribution of the sample points, is shown along with the histogram (in red) constructed with (a) the initial distribution of the sample points; and (b) the sample points after they have converged to their nearest local maximum. It should be kept in mind that the density function and the histograms wrap over at $\pm\pi$. It may be noted that the density estimate has not been normalized and does not necessarily integrate to one.

in an angular cluster measured along the direction of the cluster orientation.

Let AB and CD be two scan line segments, with mid-points M and N respectively, that belong to the same angular cluster. Let us suppose that the orientation of the cluster is aligned with the line XX' (Figure 2.4). The distance between M and N, measured along the perpendicular to XX', is defined as the lateral separation, d, between the segments AB and CD. The distance of D from A measured along the cluster orientation XX', is defined as the longitudinal overlap, p, between the segments AB and CD.

For any two scan line segments in a given angular cluster, if $d < d_{max}$ and $p > p_{min}$, where d_{max} and p_{min} are two pre-specified thresholds, we consider them to be spatially close to each other. Two scan line segments which are spatially close to each other are considered as *neighbors* and belong to the same spatial cluster. Recursive extension of this premise leads to the following criteria for a scan line segment to belong to a given spatial cluster: a scan line segment belongs to a given spatial cluster if it is a neighbor of another scan line segment that already belongs to the same cluster or if it is a neighbor of a scan line segment whose neighbor(s) already belong(s) to the same cluster. The pseudo-code listed in Figure 2.5 explains the procedure underlying the formation of a spatial cluster.

The function `spatialCluster( )` responsible for the creation of spatial clusters takes the scan line segments held by an angular cluster, say `AC`, as input. For each scan line segment `L` in the angular cluster, which has not been considered before, the

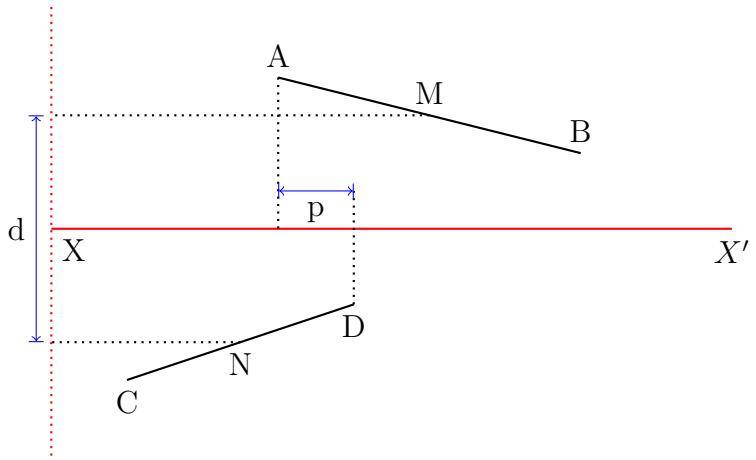


Figure 2.4: The lateral separation and longitudinal overlap between the scan line segments AB and CD are represented by d and p respectively. The cluster orientation is aligned with the line XX'.

```
1: spatialCluster(angularcluster: AC)
2: ID = 0
3: for (each line segment L in AC) do
4:     if L is not visited then
5:         mark L as visited
6:         Neighbors = findNeighbours(L, AC)
7:         ID = new Cluster ID
8:         growCluster(L, Neighbors, ID, AC)
9:     end if
10: end for
11:
12: growCluster(line segment: L, line segments: Neighbors, clusterID: ID, angularcluster: AC)
13: add L to cluster ID
14: for (each line segment L′ in Neighbors) do
15:     if L′ is not visited then
16:         mark L′ as visited
17:          add L′ to cluster ID
18:         Neighbors′ = findNeighbours(L′, AC)
19:         Neighbors = Union of Neighbors′ and Neighbors
20:     end if
21: end for
22:
23: findNeighbours(line segment: L, angularcluster: AC)
24: return (all line segments in AC which are spatially close to L)
```

Figure 2.5: Pseudo-code for the formation of spatial clusters.

neighbors of L are identified from amongst the remaining lines of AC by checking the spatial closeness criteria with L. Thereafter, L and all its neighbors are assigned to a new spatial cluster and a unique identity (ID) is assigned to the cluster. Subsequently, this cluster is allowed to grow by including all neighbors of the neighbors of L, and their neighbors and so on in turn, which are not already included in the cluster. It is important to note that *neighbor* is a transitive relation and hence the formation of spatial cluster is independent of the order in which the scan line segments are visited.

On conclusion of the formation of spatial clusters corresponding to each angular cluster, scan line segments as representatives of different linear features get segregated into different spatial clusters. We define the *support* of a spatial cluster as the number of scan line segments that belong to that cluster.

The worst case complexity of this step is $O(n_A^2)$, where n_A is the number of scan line segments in the angular cluster under consideration.

2.4.4 Estimation of the attributes of the linear features

All scan line segments in a spatial cluster are finally merged to generate a resulting line segment that captures the attributes (viz., orientation, length and position) of a distinct linear feature of the environment. All line segments, resulting from the merging of scan segments in all spatial clusters, when aggregated together produce a line segment-based map of the environment. Each such resulting segment is an instance of a *map segment*.

Let us suppose that there are n_S scan line segments in a given spatial cluster. The line supporting such a scan line segment may be represented in the normal form as (p_i, ϕ_i) where

$$x \cdot cos\,\phi_i + y \cdot sin\,\phi_i = p_i \qquad \text{for} \quad i = 1, 2, ..., n_s \tag{2.16}$$

where p_i is the perpendicular distance of the line from the origin and ϕ_i is the angle that the perpendicular from the origin on the line makes with the positive direction of the x-axis. Here, by line supporting a scan line segment, we refer to the underlying line (of infinite length) of which the scan line segment is a part.

If (p_{res}, ϕ_{res}) represent the line supporting the resultant line segment, we define

$$p_{res} = \frac{\sum_{i=1}^{n_s} p_i \cdot w_i}{\sum_{i=1}^{n_s} w_i} \tag{2.17}$$

$$\phi_{res} = atan^* \left(\frac{\sum_{i=1}^{n_s} w_i \cdot sin\,\phi_i}{\sum_{i=1}^{n_s} w_i \cdot cos\,\phi_i} \right) \tag{2.18}$$

where $atan^*(\ldots)$ is the quadrant-specific arc tangent function commonly available as the library function `atan2(...)` in most high-level programming languages and w_i is the weight associated with each scan line segment that is set to the length of the segment. Once the equation of the line supporting the resultant line segment has been obtained, it needs to be converted to a map segment by determining its two end points. The end points of the resultant map segment are obtained by projecting the end points of the constituent scan line segments of the spatial cluster on the resultant line of infinite length and choosing the end points that give the maximum length of the resultant line segment.

The scan line segments in a given spatial cluster, which are representatives of a particular linear feature, usually contain both redundant information as well as new evidence. The process of estimating the resultant line segment by merging all scan line segments in a spatial cluster is the way by which we combine all line segments that represent the same linear feature into a single line segment. The computational complexity of this step is $O(n_s)$

2.5 Discussion

Our literature survey revealed that two methods—one proposed by Amigoni and Vailati [74] and the other by Lakaemper [75]—come close to the one proposed in this Chapter. But the method of Lakaemper [75] has some crucial differences with the one proposed here. In Lakaemper's method [75], line segments are extracted from pre-aligned scans having orientation in the range $[0, \pi]$. This leads to loss of vital information concerning the sense in which the LRF had scanned the corresponding obstacle surface. In the proposed method, we extract scan line segments in the range $(-\pi, \pi]$ and thereby retain this information. The senses of orientation of the map segments simplifies implementation of localization algorithms, such as [128], that work by matching scans to map segments, because the task of establishing correspondences becomes relatively straightforward then. To illustrate this point through a contrived example, let us suppose that the robot is somewhere in the vicinity of a thin rectangular object QRST housed in a rectangular room (Figure 2.6) and that it needs to update its pose. If the orientations of the segments QR and ST are known—the two orientations are opposite—the robot can easily match its laser range scan with the correct segment (either QR or ST) as the direction of the scan and the orientation of the corresponding segment would be identical. If the orientations of the scan line segments (and hence the directedness of map segments) are sacrificed, establishing the correspondence would be tricky and not so straightforward. Hence, it is crucial to retain the identity of the opposite faces of even very thin objects that the robot may possibly see using map segments of opposite orientations. Furthermore, extracting scan line segments in the $(-\pi, \pi]$ range prevent

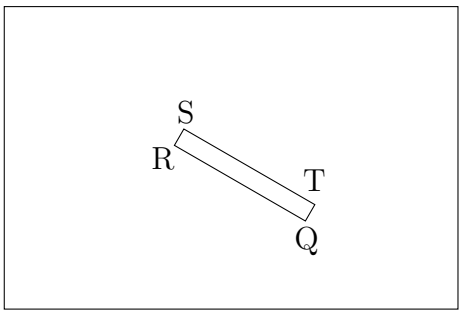


Figure 2.6: Preservation of the senses of orientation of the map segments simplify localization. Moreover, closely-lying but oppositely oriented segments, like QR and ST here, do not get merged and model two opposite faces of an object distinctly.

closely spaced line segments like QR and ST in Figure 2.6 from getting merged because the two line segments have opposite senses of orientations and hence will belong to two different angular clusters. The issue of sense of orientation in the map segments has also been considered in [116, 118]. We assert that the sense of orientation of the map line segments is a vital piece of information that we can ill-afford to ignore and hence advocate representing maps by directed line segments, as long as they are built from range data acquired from LRFs.

In addition to the above, the method of Lakaemper [75] considers the distance between line segments in the *centre-direction* joint space. To put the spatial difference and angular difference on an equal footing, a spatial difference of 200mm is considered equivalent to 10° and line segments exceeding a certain length are split into smaller segments. This, in our opinion, is subjective and arbitrary. While we always strive to extract long line segments and assign more weights to longer segments as they are perceptually more significant, this method splits up long segments into shorter ones so that all segments get similar importance. This appears counter-intuitive. However, the advantage that accrues with this formulation is that it allows line segments belonging to the same cluster to differ more widely in direction if they are spatially close and vice-versa. In our method, in the first clustering step (using the mean-shift procedure) we use only the orientations of the scan line segments and the distance between two segments are computed using a circular measure since orientations are directional data. In the second clustering step, we consider only the spatial attributes. At both the stages, the data corresponding to the line segments are weighted by the lengths of the segments.

Compared to the method of Amigoni and Vailati [74], which uses a hard-coded threshold, ε, to bring together scan line segments for merging irrespective of their orientations, our method uses a somewhat adaptive approach. Once our method has identified closely oriented scan line segments, all "neighboring" scan line segments are brought together for merging. By using a transitive relation for identifying the

neighbors, the ambit of neighborhood is allowed to grow based upon the input data. As we shall see in the following Chapter, experiments were performed with different datasets (public, real and simulated) and the same setting of the parameter that defines an initial tentative neighborhood, was found to give good results for all cases.

Chapter 3

Assessment of line segment-based maps

3.1 Introduction

In order to establish the efficacy of a given mapping method, it is imperative to assess the goodness of the maps produced by the method. Unfortunately, maps generated by most of the methods proposed in the literature have not been assessed. Only pictorial depictions of the maps have been made available for visual inspection [74, 75, 87, 117–121]. This lacuna on map assessment can possibly be attributed to the absence of a general agreement in the robotics community as to what constitutes a reasonable yardstick for map assessment. Formulation of such a yardstick is indeed a daunting task as the desirable attributes of a map are often conflicting and are guided by the application in which the map is to be used. For path-planning, it may be desirable that even small objects in the environment like legs of tables are modeled in the map, while for localization it may be desirable to have relatively long objects only, but their positions must be known very accurately. Also, the difficulty in obtaining the ground truth precludes the possibility of quantitatively determining how closely a map models a given environment. In [129], wherein an on-line method for segment-based map building from sonar data was proposed, the generated maps were compared with the real maps using a mean absolute error criterion. Lakaemper [130] proposed a method for evaluating line segment-based maps by defining a *confidence measure* that does not take into account the ground truth. Unfortunately, the method fails to guarantee that a map with a high confidence models the environment correctly. A framework for benchmarking SLAM methods, which may differ in their sensing modalities as well as in map representations, was proposed in [131].

Amigoni et al. [132] made a critical review of the commonly-adopted practices in the experimental activities on line segment-based mapping before recommending a number

of issues to be followed for experimentally validating a mapping method. In their view, adherence to these recommendations by researchers will open up the possibility of evaluating and comparing different mapping methods by ensuring repeatability of the experiments. Following their recommendations, in our experimental activity we have: (i) applied our method on data available in the public domain; (ii) compared our results with that of two other methods on public data; (iii) indicated the sizes of mapped environments; (iv) indicated the number of line segments in each map; (v) indicated the displacements between the scans; (vi) evaluated the maps against ground truth (wherever available) both pictorially as well as quantitatively; (vii) mentioned the values of the important parameters; (viii) used closed loop paths; and (ix) indicated the processing time. Additionally, the line segments of the maps generated by our method are directed and preserve the sense in which the corresponding linear surfaces have been scanned by the LRF. We also (a) indicate the average displacements between two consecutive poses as well as information related to the lengths of the map segments; and (b) analyze how the results of our method change when the parameter settings of our method are perturbed.

3.2 Experimental evaluation

Before proceeding to present the experimental results, it is important to point out that the scan poses are generally estimated through odometery. Before long, such estimates turn out to be unreliable because of cumulative drift errors. This leads to relationships among the scan poses that are globally inconsistent. Hence, the need arises to obtain improved estimates of the scan poses to make the relationships globally consistent so that all scan data are consistently aligned. One way of achieving this is through the use of a *full* SLAM method that estimates the entire trajectory traversed by the robot along with the map. A popular paradigm for solving the full SLAM problem is based on a graphical formulation of the problem and then solving it offline using nonlinear least squares method. A seminal work that provided the first working solution in this paradigm is attributable to Lu and Milios [71]. In later years, many refinements and improvements to this work have been reported in literature e.g., [133–139]

In the Lu-Milios method [71], the poses of the robot are represented as nodes of a graph [140], while the spatial relationships between poses resulting from pair-wise matching of laser range scans or from odometry are represented by arcs. Once such a graph is constructed, the problem reduces to one of finding the configuration of the nodes that maximizes the likelihood of the measurements contained in the arcs. The problem is then solved simultaneously for all the nodes, by minimizing the Mahalanobis distance between the observed and derived relations over the entire graph. A major drawback of this method is that as the number of scan poses S increases, the complexity

of the algorithm grows as $O(S^3)$.

If we assume that the robot takes a laser range scan S_0 at its initial pose X_0 and for each subsequent step of movement from pose X_{i-1} to X_i, the robot takes a new laser range scan S_i at the new pose X_i, the broad steps of the Lu-Milios algorithm [71] can be summarized as follows:

1. Each previous scan $S_j,\ j = 0, 1, \ldots, i-1$ is matched with the current scan S_i to estimate the pose relation between X_i and X_j as $D_{ij} = X_i - X_j$. The observation D_{ij} is modeled as $\widetilde{D}_{ij} = D_{ij} + \triangle D_{ij}$, where $\triangle D_{ij}$ is a random Gaussian error with zero mean and known covariance matrix C_{ij}.

2. If we suppose that $\boldsymbol{X}$ represents the concatenation of optimal pose estimates $X_0, X_1, \ldots, X_n$, where $X_0 = [0, 0, 0]^T$, $\boldsymbol{D}$ the concatenation of all the relations of the form $D_{ij} = X_i - X_j$, and $\boldsymbol{H}$ the incidence matrix such that $\boldsymbol{D} = \boldsymbol{HX}$ then $\boldsymbol{X} = [\boldsymbol{H^t C^{-1} H}]^{-1} \boldsymbol{H^t C^{-1}} \widetilde{\boldsymbol{D}}$ and $\boldsymbol{C_X} = [\boldsymbol{H^t C^{-1} H}]^{-1}$ where $\widetilde{\boldsymbol{D}}$ is the concatenation of all observations $\widetilde{D}_{ij}$ for the corresponding D_{ij} and $\boldsymbol{C}$ is the covariance of $\widetilde{D}_{ij}$.

In our experimental activities, whenever we confront input scan data in which the scan poses are estimated through the error-prone odometry, we get the input data pre-processed through the Lu-Milios algorithm [71] to obtain improved estimates of the poses. If, however, corrected scan poses are already available with the input data, we do not subject the data to preprocessing and instead feed it directly to our method. It is important to note that we use the Lu-Milios method [71] only to perform a pre-processing on the input dataset to obtain better estimates of the scan poses and thereby align the scans. It is not intrinsic to the proposed method, and in principle, any full SLAM algorithm that works using wheel odometry and laser range scan data could be used in its place. However, the output generated by the proposed map building method is affected by how well the scan registration method succeeds in obtaining a globally consistent pose estimate.

From the standpoint of representing the ground truth both accurately as well as with fewer line segments, it is important that the scan registration method is accurate. If the estimates of the scan poses provided by the scan registration algorithm are accurate, the line segments of the map would model the ground truth closely, provided of course that the scan acquisition is accurate and the line merging process does not introduce too large an error. To better understand the dependence of the number of line segments in a map on the accuracy of the scan registration process, let us imagine a situation where a given linear feature is observed in several scans, all collected from different scan poses. Since the scan poses are estimated through odometry, the scans will in general be grossly misaligned. An effective and accurate scan registration method

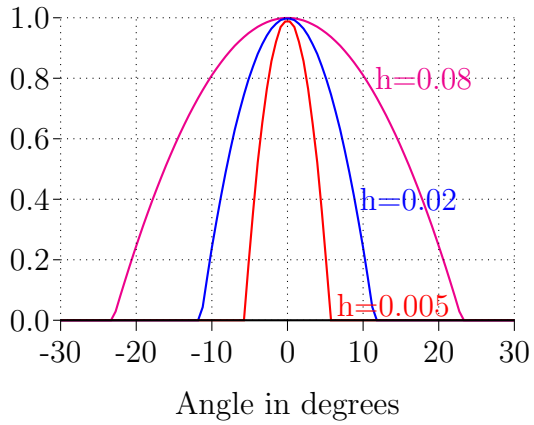


Figure 3.1: The biweight kernel of equation (2.13) for three different values of the kernel viz., $h = 0.005$, $h = 0.02$ and $h = 0.08$.

would align the scans appropriately and would permit the use of a single line segment to faithfully model the linear feature. An ineffective scan registration method would fail to align the scans appropriately. Thus, we may end up with having several line segments (improperly) representing the linear feature.

Our experimental activity can be broadly divided under three categories. First, to show the efficacy of our method in mapping large indoor environments, we have applied our method on three different datasets available publicly in the *Robotics Data Set Repository* (Radish) [141]. Second, we assess the map of a medium-sized indoor environment against ground truth data, where the data have been manually extracted. Third, we test our method in a large simulated environment whose ground truth is accurately known. However, before describing the experimental results, we shall digress briefly to discuss on the important parameters of our method and their settings.

The parameter that regulates the formation of the angular clusters and has to be set judiciously is the bandwidth, h, of the kernel function. A large bandwidth leads to an over-smoothed probability density function with fewer modes, while too small a bandwidth yields a noisy density function that elicits more modes than are actually present in the data. Figure 3.1 shows the kernel function represented by equation (2.13) for three different values of the bandwidth. In all the experimental results given in this Chapter, the bandwidth was kept fixed at $h = 0.02$. This setting corresponds to a base spread of around 11.5° on either side of the center (see Figure 3.1, blue curve).

Two parameters control the formation of the spatial clusters—the lateral separation, d_{max} and the longitudinal overlap, p_{min}. If too large a value of d_{max} is chosen, a *laterally fat* spatial cluster results. Such a cluster merges similarly oriented but distantly placed distinct linear features and prevents them from standing out separately. Too small a value of d_{max} results in more than one line segment representing the same linear feature in the map because they could not be merged when they should have been. A

linear feature often finds itself represented by two or more collinear (or nearly so) line segments because of undesirable breaks in scan line segments resulting from spurious occlusions or noise in the range data. A negative value of the parameter, p_{min}, helps join all such breaks. In essence, the longitudinal overlap, p_{min}, regulates to what extent a spatial cluster can elongate itself. In all the experiments described in this thesis, these two parameters were set as: $d_{max} = 400mm$ and $p_{min} = -100mm$.

The parameter that has a significant bearing on the processing time of the proposed method is the convergence criterion of the mean-shift clustering procedure. In our experiments, we had considered the procedure to have converged if the values of any of the sample data points did not change by over 5° from one iteration to the next. Setting this value to 1° did not change the final result in most of the cases, and only in a minor way in the remaining of the cases, but increased the running time by 1.5–3 times by necessitating more number of iterations for the method to converge.

Two parameters that regulate the granularity of representation in the map and also help eliminate spurious line segments arising from dynamic and irrelevant objects are the *minimum line length threshold*, L_{min} and the *minimum support threshold*, S_{min}, of the spatial clusters. If a linear feature is not sufficiently long (say, smaller than L_{min}), it may not be worthwhile to retain the feature in the map as it adds only to the cost of map storage without contributing much valuable information. If, too large a value for L_{min} is chosen, we may end up with dropping important features from the map. If a linear feature is of transient nature, it will not be observed in too many scans at the same place. Thus, it will be represented by several separate spatial clusters but none of them will have too many scan line segments belonging to them. Thus, by discarding all spatial clusters whose support does not exceed a chosen S_{min}, we eliminate dynamic features from the final map. However, the choices of L_{min} and S_{min} are somewhat interrelated. In general, a linear feature, which is short in length, will be observed from only a few scan poses while a long feature will be visible from many. Thus, the support of the spatial cluster representing a short linear feature will naturally be small as compared to that of a long feature. Thus, to be effective, S_{min} should be small when L_{min} is small and large when L_{min} is large. The value of S_{min} is also dependent on how closely range scan data have been collected. In our experiments, we have used the following setting of these parameters: $L_{min} = 500mm$; $S_{min} = 5$.

3.2.1 Mapping with data available in the public domain

In the first experimental exercise, we apply our method on the following three indoor datasets obtained from Radish repository [141]:

(a) A 375m long corridor of the main building of Chosun University;

(b) Department of DIIGA, Engineering University, Ancona (∼size: 47m×47m); and

(c) SRI AIC K wing (∼size: 83m×21m).

For the dataset (a), the scan poses were estimated through odometry; hence, the Lu-Milios method [71] was applied on the dataset before invoking our method. For, the remaining two datasets, this pre-processing step was not necessary as the scan data were already pre-registered using scan matching techniques [141].

The dataset (a) contains 23,067 scans and 4,175,127 scan points. Among these, as many as 9423 scans were taken from poses that were identical to their immediate preceding poses. The dataset (b) contains 8540 scans and 1,539,349 scan points. Here, there are as many as 7671 cases where the displacements between two consecutive scan poses differ by less than 100mm and 5°. These indicate that there are significant amounts of redundant data in the two datasets. With a view to eliminating redundant data, we processed only those scans for which two consecutive scan poses differed either by at least 250mm or by at least 10°. The displacement between two consecutive poses in dataset (c)—which contained 1421 scans and 244,614 scan points—were much larger and in only one case it was found to be less than 250mm and 10°. Figure 3.2 shows the scan points as contained in the dataset after removal of redundant data and the line segment maps extracted by the proposed method. Figure 2.3 in page 25 pertains to the mean-shift clustering step applied on dataset (c).

In Table 3.1 we enumerate a few attributes of the datasets. In column 2, the numbers of scans that have been processed (left after removal of redundant scans) for extraction of the line segment-based maps are shown. The average displacements between two consecutive scan poses of these processed scans are indicated in columns 3 (translational displacement) and 4 (angular displacement). Column 5 lists the numbers of scan line segments extracted from the datasets using the step detailed in Section 2.4.1 on page 21, while the next two columns indicate respectively the lengths of the smallest and the largest scan line segments so extracted. Column 8 shows the average length of the scan line segments.

Table 3.1: Attributes of datasets after removal of redundant scans and that of the scan line segments extracted from the scans.

Dataset	#Scans proc.	Avg. Δt(mm)	Avg. Δa	#Segments	Smallest(mm)	Longest(mm)	Avg. length(mm)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Dataset (a)	1307	14	0.08°	6187	100	4213	952
Dataset (b)	1289	32	0.89°	7066	100	5571	849
Dataset (c)	1420	469	12.30°	8374	100	5524	713

Table 3.2 shows the attributes of the maps generated by the proposed method from the datasets listed in column 1. Column 2 indicates the number of line segments in the final map. Columns 3 and 4 indicate respectively the lengths of the smallest and the largest line segment in the generated map and column 5 indicates the average length of the map segments. The processing time required to compute the maps, starting from

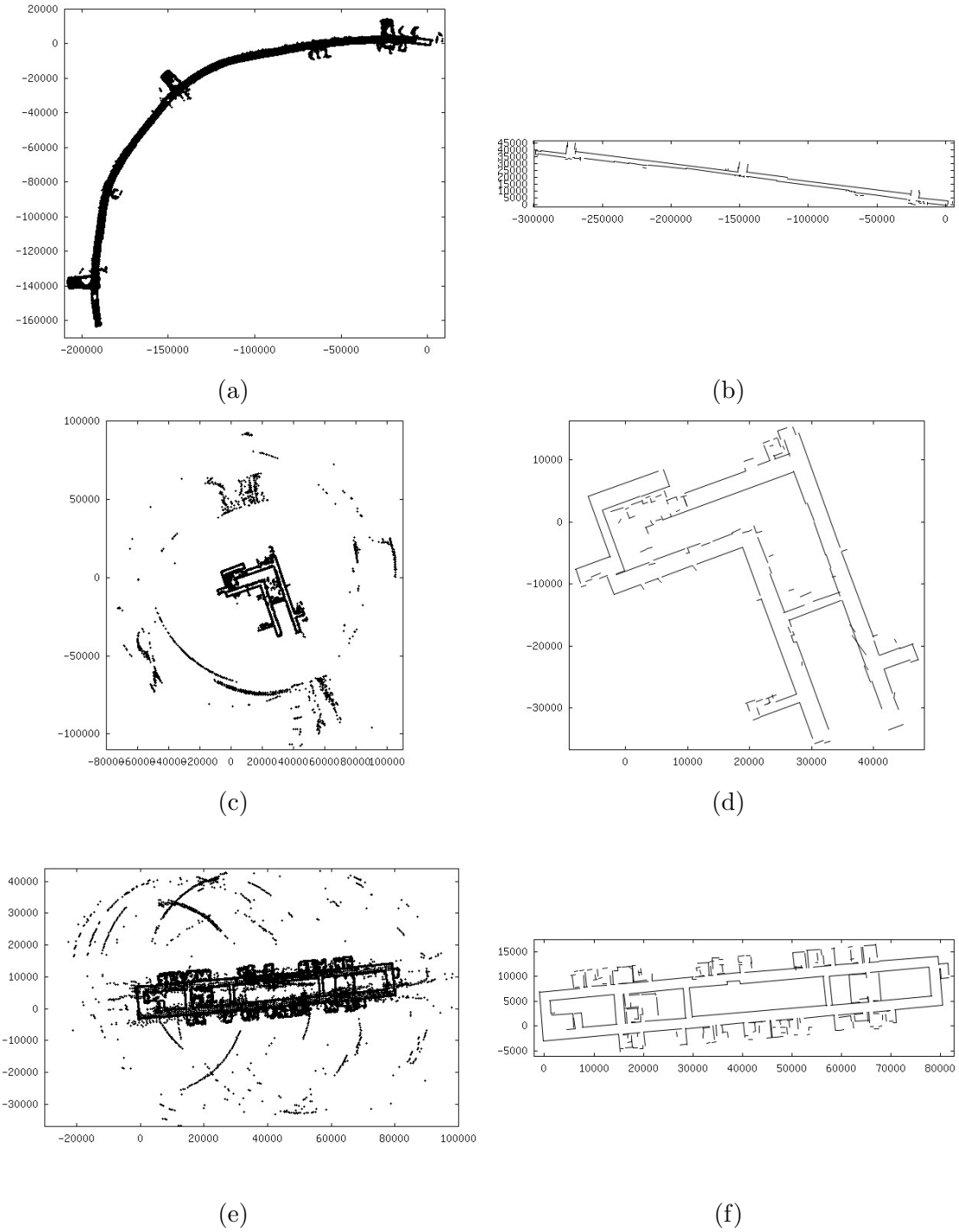


Figure 3.2: Figures (a), (c) and (e) on the left-hand side show laser scan points as contained in the datasets after removal of redundant scans, while Figures (b), (d) and (f) on the right-hand side show the corresponding line segment maps extracted by the proposed method from datasets (a)—Chosun University; dataset (b)—Department of DIIGA; and dataset (c)—SRI AIC K wing, respectively. All dimensions are in mm. The scans in dataset (a) had to be pre-aligned using the Lu-Milios [71] method.

reading in the (registered) range scan data as input, to writing out the end points of the constituent directed map segments, is listed in column 6.

Table 3.2: Attributes of the maps produced by the proposed method.

Dataset	#Map segments	Smallest segment (mm)	Largest segment (mm)	Average length (mm)	Proc. time (s)
(1)	(2)	(3)	(4)	(5)	(6)
Dataset (a)	75	502	119,681	8502	107
Dataset (b)	143	502	22,483	2857	143
Dataset (c)	237	508	27,326	2594	340

In order to see how the maps produced by the proposed method compare with maps generated by other methods proposed in the recent past, we enumerate in Table 3.3 the different attributes of the maps produced by the methods due to Amigoni and Vailati [74] and to Lakaemper [75]. These methods are invoked on the scan line segments referred to in column 5 of Table 3.1. Thus all the three methods work with the same sets of scan line segments and the differences in the final results are attributable only to the intrinsic differences in the methods. Since all the three methods under consideration have one or more parameter(s) on which the maps produced are dependent, it is imperative to set them judiciously so that a fair comparison is possible. It has already been pointed out in page 16 (Section 2.2) that the method [74] uses a parameter ε to determine which scan line segments would be brought together for possible merging. Since there is no parameter in our method which directly relates to ε, we had estimated an equivalent value of ε (we call it ε_1) corresponding to the maps generated by our method. Subsequently, we computed the maps by [74] for $\varepsilon = \varepsilon_1$. To arrive at a plausible value of ε_1, the end point of the line segment in a spatial cluster, which is at a maximum distance from the resultant map segment arising from the cluster (on either side of the map segment), is identified. The two end points (one on each side of the resultant map segment) define a strip around the resultant segment, on or inside which all line segments of the spatial cluster which underwent merging, reside. The widths of all such strips, corresponding to all spatial clusters, are computed. Of all such strips, the one that has the maximum width is then identified and the maximum width is considered equal to $2\varepsilon_1$. For example, for the map produced by our method for dataset (a), the maximum width of the strip turned out to be 2816mm. Hence ε_1 was set to 1408mm. We had also determined the value of ε (we call it ε_2) through trial and error using which value Amigoni and Vailati [74] produces the same number of map segments as our method. As far as the method [75] is concerned, it has one parameter σ that has to be set to *the minimum size of significant detail* [75]. Since no parameter of our method had any direct bearing with this parameter, we generated maps by this method for widely differing values of σ.

From Table 3.3, it is seen that with datasets (a) and (c), for the same strip width, the maps produced by our method contain fewer segments, and are thus more compact, than those produced by the method of Amigoni and Vailati [74]. Alternatively, for the same number of map segments produced by the two methods, in our method,

Table 3.3: Attributes of the maps produced by Amigoni and Vailati [74] and Lakaemper [75].

Method	Dataset	Parameter (mm)	#Map segments	Smallest segment (mm)	Largest segment (mm)	Average length (mm)
(1)	(2)	(3)	(4)	(5)	(6)	(7)
Amigoni and Vailati [74]	Dataset (a)	$\varepsilon_1 = 1408$	135	117	75,076	4474
		$\varepsilon_2 = 2495$	75	102	87,052	5365
	Dataset (b)	$\varepsilon_1 = 664$	128	106	20,917	2944
		$\varepsilon_2 = 500$	143	108	20,917	2691
	Dataset (c)	$\varepsilon_1 = 489$	264	0	27,325	2038
		$\varepsilon_2 = 550$	237	0	27,325	2199
Lakaemper [75]	Dataset (a)	$\sigma = 500$	1643	101	1,812	749
		$\sigma = 1000$	1239	101	2,723	1056
		$\sigma = 5000$	1112	101	4,213	1335
		$\sigma = 10,000$	1162	101	4,213	1335
	Dataset (b)	$\sigma = 500$	877	103	2,495	762
		$\sigma = 1000$	868	103	3,144	980
		$\sigma = 5000$	1038	103	4,598	1203
		$\sigma = 10,000$	1028	103	4,598	1147
	Datatset (c)	$\sigma = 500$	1478	100	1,692	650
		$\sigma = 1000$	1635	102	2,586	744
		$\sigma = 5000$	1650	103	4,117	838
		$\sigma = 10,000$	1611	101	4,117	837

the scan line segments that give rise to the final map segment lie closer to the final segment. For instance, for dataset (a), with a strip width of 1408mm, our method produces 75 map segments while Amigoni and Vailati [74] produces 135 segments; to produce 75 segments, Amigoni and Vailati [74] requires a strip width of 2495mm. Thus, the maps produced by our method are more compact and accurate. Amigoni and Vailati [74], however, outperforms the proposed method for dataset (b) on the above considerations. As far as the lengths of the map segments are considered, the proposed method generates maps with segments having lengths larger than (or similar to) the lengths of map segments produced by Amigoni and Vailati [74]. For dataset (a), the map produced by our method contains substantially fewer but longer segments than the method of Amigoni and Vailati [74]. Close visual scrutiny revealed that the map produced by method of Amigoni and Vailati [74] has quite a few map segments that are collinear (or nearly so), with small gaps among them along their lengths. Since our method has a mechanism—achieved through the use of a negative value of the parameter p_{min}—to join all such gaps between collinear map segments, it produced fewer and longer map segments.

If we augment the methods of Amigoni and Vailati [74] and Lakaemper [75] by adding a step that discards from the maps produced by them those line segments whose lengths do not exceed L_{min}, we get fewer map segments than what is reported in column 4 of Table 3.3. Even after discarding, the number of map segments that remain for the method of Lakaemper [75] is substantially higher than that of the proposed method and that of Amigoni and Vailati [74]. With the method of Amigoni and Vailati [74], after the line segments have been so discarded, maps contain fewer segments than those produced by the proposed method for dataset (b) and (c). However, for dataset (c), the scan line segments that give rise to the final map segment lie closer to the final segment as in dataset (a) for the proposed method and thus the map produced by it

can be considered to be more accurate.

The method of Lakaemper [75], however, produces maps with a substantially higher number of segments with shorter lengths compared to the maps produced by Amigoni and Vailati [74] and the proposed method. Visual inspection of the maps produced by Lakaemper [75] revealed many closely lying line segments, the kind of which were merged by Amigoni and Vailati [74] and the proposed method but not by Lakaemper [75].

To see the dependence of the attributes of the maps produced on the parameters of our method, we plot in Figure 3.3 the variation in the different attributes of the maps listed in Table 3.2 as the parameters are varied. The results for dataset (a) only are shown; similar plots are obtained for the other datasets. In Figures 3.3a–3.3f, when one parameter is varied, the other parameters are kept fixed at their stipulated values, specified earlier. When h is small (Figure 3.3a), a large number of angular clusters are formed. Since each of these clusters gets split into one or more spatial clusters, a large number of map segments are obtained. When d_{max} is small (Figure 3.3b), a large number of slender spatial clusters are formed. As d_{max} is increased, fewer spatial clusters which are spatially fat result. Hence, the number of map segments decreases with increasing d_{max}. With increase in d_{max}, the cluster support, and hence, the cluster span increases. Thus, the lengths of the map segments increase with increasing d_{max}. With the increase in the negative value of p_{min} (Figure 3.3c), more and more collinear (or nearly so) map segments with wide gaps along their lengths start getting merged, resulting in longer but fewer map segments. As L_{min} is increased (Figure 3.3d), smaller map segments get discarded from the map. Hence, the number of map segments decreases and the length of the smallest map segment increases. As S_{min} increases (Figure 3.3e), spatial clusters with smaller support get discarded; hence, the number of map segments decreases. But with increase of S_{min}, the average support of the spatial clusters increases. Hence, the average lengths of the map segments increase. As far as the processing time is concerned, it varies significantly with h (Figures 3.3f) and only in a minor way with d_{max}. It is by and large unaffected by the remaining three parameters. For small values of h, the sample points lie on or very close to their corresponding local maxima on the probability density curve and thus convergence takes place within a few iterations. For larger values of h, a larger number of iterations become necessary for convergence, leading to higher processing time.

3.2.2 Mapping of real environment

In this experimental exercise, we manually drove a Pioneer-3DX mobile robot in a rectangular room of approximate dimensions $11m \times 6m$ while it acquired range data with a SICK LMS 200 LRF mounted on it. The room contained a rectangular duct, two

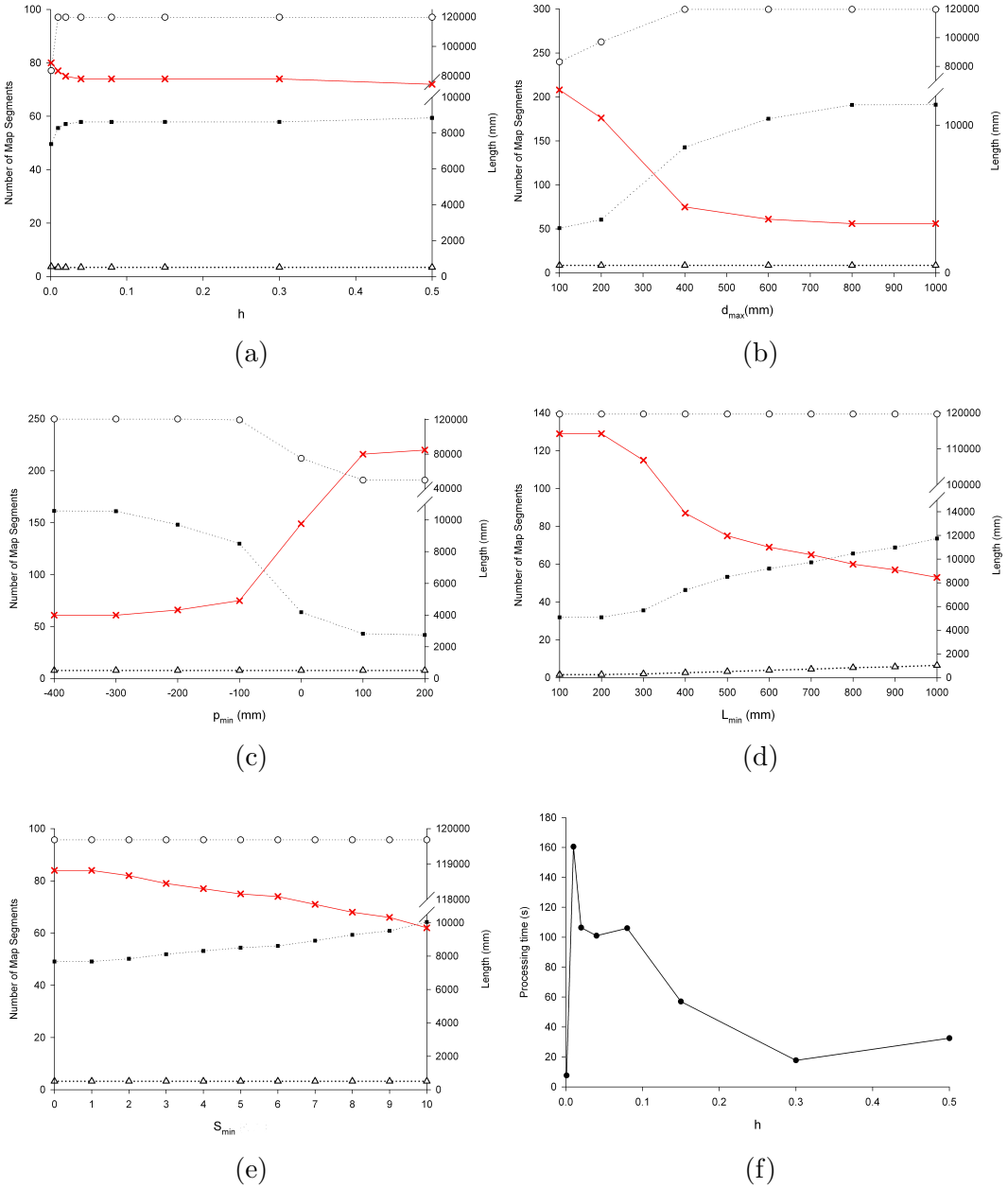
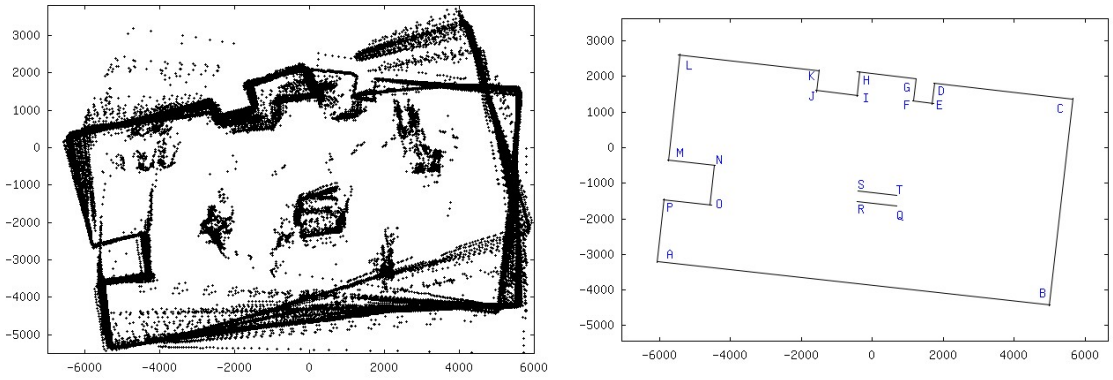


Figure 3.3: The dependence of the different attributes of the maps produced by the proposed method on the parameters of the method for dataset (a). In each of the Figures (a)–(e), the left vertical axis represents the numbers of map segments, whereas the right vertical axis represents the lengths of the map segments in millimeters. The cross sign ($\times$) is used for the number of map segments whereas the hollow circle ($\circ$), hollow triangle ($\triangle$) and filled square ($\blacksquare$) indicate the length of the largest map segment, length of the smallest map segment and the average length of the map segments respectively. The vertical axis in Figure (f) represents the processing time in seconds.The results quoted in this thesis correspond to the following settings of the parameters: $h = 0.02$; $d_{max} = 400mm$; $p_{min} = -100mm$; $L_{min} = 500mm$; and $S_{min} = 5$.

rectangular obstacles on the two sides of a closed door, a thin rectangular obstacle near the center of the room, a few chairs, a table and a human being walking around. The

consecutive scan poses differed by at least 250mm or 10° while the robot was capturing range data. The average translation and angular displacement between two consecutive poses were 200mm and 7.26° respectively. The proposed method was invoked on the laser range data collected by the robot after registering the scans using the Lu-Milios method [71]. Figure 3.4a shows the scans obtained from 154 poses, which were estimated using odometry while Figure 3.4b shows the produced line segment map. The CPU time consumed to extract the map was 1.30s.



(a) (b)

Figure 3.4: Mapping of a room of dimensions 11m×6m (a) Laser range data captured from odometry-estimated scan poses; (b) Line map containing 18 segments produced by the proposed method. The scans had to be pre-aligned using the Lu-Milios method [71]. All dimensions are in mm.

It is important to note that the proposed method succeeds in preventing scan points arising from transient and small obstacles like human legs from contributing any line segment in the final map. The two sides (left and right) of the obstacle lying near the center of the room (represented as QRST in Figure 3.4b) also find no representation. This is because the sides are 300mm wide and we had decided to retain segments in the map which are more than 500mm in length.

The dimensions of the different linear features present in the map and indicated by pairs of letters in Figure 3.4b are shown in Table 3.4. For instance, AB indicates the lower wall of the room shown in Figure 3.4b. Against each such dimension (which is essentially the length of the line segment representing the corresponding feature in the map) is shown the actual dimension of the feature as measured from the real environment. These measurements were done manually using a measuring tape and are accurate to a centimeter.

In order to assess how closely the dimensions of the objects in the actual environment are reflected in the map, we introduce an error measure, called the *dimensional*

Table 3.4: Comparison of the lengths of the map segments with the dimensions of the actual objects they represent. The actual dimensions were obtained through manual measurements at centimeter level resolution.

Feature	Map segment length (cm)	Measured length (cm)	Feature	Map segment length (cm)	Measured length (cm)	Feature	Map segment length (cm)	Measured length (cm)
AB	1111.8	1112	GH	169.2	162	MN	134.6	131
BC	585.9	581	HI	69.7	63	NO	114.7	110
CD	391.7	387	IJ	118.5	114	OP	137.1	131
DE	60.3	53	JK	62.0	55	PA	180.1	177
EF	58.3	53	KL	400.1	395	QR	111.4	107
FG	62.0	60	LM	301.1	297	ST	110.7	107

error, ε_D, in a map defined as

$$\epsilon_D = \frac{1}{M} \sum_{i=1}^{M} \frac{\mid L_i^{(m)} - L_i^{(g)} \mid}{L_i^{(g)}} \tag{3.1}$$

where $L_i^{(m)}$ is the length of the ith map segment representing a linear feature whose real length is $L_i^{(g)}$ and M is the total number of line segments in the map.

The dimensional error of the map represented in Figure 3.4b, computed in accordance with equation (3.1), is 0.047. This implies that, on an average, for every unit length of a given dimension of an obstacle in the actual environment, the length of the corresponding map segment departs from the true dimension, by about 4.7%. However, it must be kept in mind that the heights above the ground at which the manual measurements were made were different from the height at which the LRF scanned the environment. Since, one cannot assert that all object dimensions remain unchanged in the vertical direction, one must also take this factor into account when considering the dimensional error.

3.2.3 Mapping in simulated environment

We also applied our method on laser scan data collected from a simulated environment. The environment had the shape of a regular octagon and had a square obstacle at the center. All sides of the octagon and the square obstacle were 10m in length. Laser range data were taken from 105 poses by driving a simulated robot around the square obstacle. Consecutive poses differed by at least 1m or 10°. (Since the environment was fairly uniform, we decided to take scans after translations of 1m in place of 250mm.) The average pose displacements between two consecutive poses were 631mm and 4.86°. We mimicked odometry in the simulated environment by injecting Gaussian noise to the true pose estimate obtained from the simulator. During the data collection phase, the odometry estimate of a scan pose $(\widehat{x}_i, \widehat{y}_i, \widehat{\alpha}_i)$ was obtained from its preceding odometry-estimated scan pose $(\widehat{x}_{i-1}, \widehat{y}_{i-1}, \widehat{\alpha}_{i-1})$ by injecting Gaussian noise to the difference between the two corresponding true poses obtained from the simulator,

as shown below.

$$\widehat{x}_i = \widehat{x}_{i-1} + \widehat{d}_i \cdot cos\left(\widehat{\alpha}_{i-1} + \frac{\widehat{\theta}_i}{2}\right) \tag{3.2}$$

$$\widehat{y}_i = \widehat{y}_{i-1} + \widehat{d}_i \cdot sin\left(\widehat{\alpha}_{i-1} + \frac{\widehat{\theta}_i}{2}\right) \tag{3.3}$$

$$\widehat{\alpha}_i = \widehat{\alpha}_{i-1} + \widehat{\theta}_i \tag{3.4}$$

where

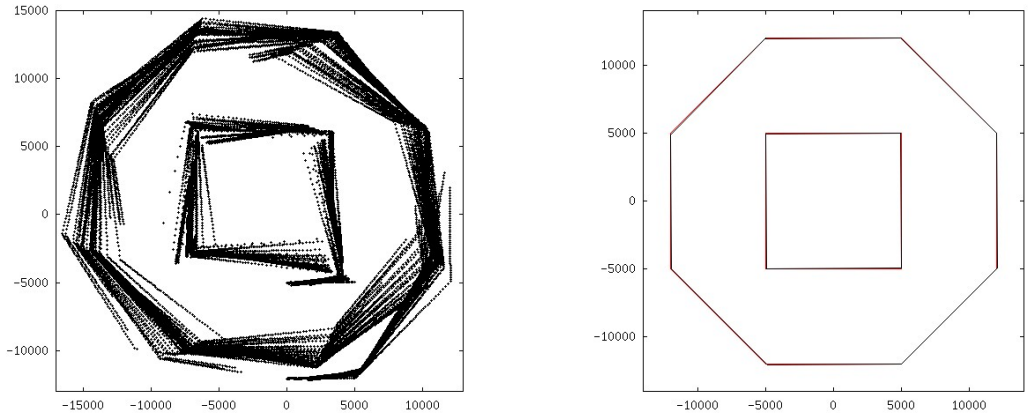
$$\widehat{d}_i = d_i + d_i \cdot N\left(0, \sigma_{\Delta d,d}\right) + \theta_i \cdot N\left(0, \sigma_{\Delta d,\theta}\right) \tag{3.5}$$

$$\widehat{\theta}_i = \theta_i + \theta_i \cdot N\left(0, \sigma_{\Delta\theta,\theta}\right) + d_i \cdot N\left(0, \sigma_{\Delta\theta,d}\right) \tag{3.6}$$

In the above equations, d_i and θ_i are the translation and the orientation change respectively between the $(i-1)$th and ith true pose obtained from the simulator. $N(0, \sigma)$ indicates zero-mean Gaussian noise with standard deviation σ. The Gaussian noise parameter $\sigma_{\Delta d,d}$ is the standard deviation of the error in translation arising per unit of translation and $\sigma_{\Delta d,\theta}$ is the standard deviation of the error in the translation per unit change in the orientation of the robot. The other two noise parameters $\sigma_{\Delta\theta,d}$ and $\sigma_{\Delta\theta,\theta}$ have analogous implications for orientation. The following values for these parameters were chosen: $\sigma_{\Delta d,d} = 20\text{mm/m}$, $\sigma_{\Delta d,\theta} = 0.1\text{mm}/1^\circ$, $\sigma_{\Delta\theta,\theta} = 4^\circ/360^\circ$ and $\sigma_{\Delta\theta,d} = 4^\circ/\text{m}$.

Figure 3.5a shows the laser scan data obtained from poses estimated using the above procedure. The scans were then aligned using the Lu-Milios method [71] and then the proposed method invoked on the data. Figure 3.5b shows the extracted map (in black) superimposed on the ground truth (in red). Since, the difference between the extracted map and the ground truth is very small compared to the dimensions of the environment, the difference is hardly visible. The CPU time consumed to extract the map was 0.10s. The shortest, longest and the average lengths of the extracted map segments are 9921 mm, 10,051 mm and 9990 mm respectively. The dimensional error of the map is 0.004.

The availability of precise dimensions and locations of the obstacles enabled us to assess how far the generated map departed from the ground truth. The assessment



(a) (b)

Figure 3.5: (a) Scan data collected from 105 poses in the simulated environment; (b) The generated map (in black) superimposed on the ground truth (in red). All dimensions are in mm.

was done separately through the use of *Hausdorff Distance* (HD) and the *Oriented Segment Hausdorff Distance* (OSHD) [142]. This may be thought of as an alternative error measure, where the correspondences of the map segments to the real objects need not be known. In the first case, we assume that the end points of the line segments representing the ground truth and the map belong to the two sets, T_p and M_p, respectively. In the second case, two sets, designated as T_s and M_s respectively, contain the segments representing the ground truth and the map.

The one-way HD distance between T_p and M_p is given as

$$h(T_p, M_p) = \max_{a \in T_p} \left(\min_{b \in M_p} \left(d\left(a, b\right) \right) \right) \tag{3.7}$$

where $d(a, b)$ is the Euclidean distance between the points a and b.

The one-way OSHD between two sets T_s and M_s consisting entirely of line segments is defined as [142]

$$h_{OS}(T_s, M_s) = \max_{u \in T_s} \left(\min_{v \in M_s} \left(max\left(d(u^s, v^s), d(u^e, v^e) \right) \right) \right) \tag{3.8}$$

where the superscripts s and e associated with the line segments u and v indicate their start and end points. The point from where the LRF starts scanning a linear feature is the start point of the corresponding representative segment. The definitions of $h(M_p, T_p)$ and $h_{OS}(M_s, T_s)$ are analogous.

The OSHD [142] addresses the issue of correspondence when assessing the disparity between two sets of line segments, whereas the (standard) HD computed between two

Table 3.5: Distance between the produced map and the ground truth computed using HD and OSHD.

Hausdorff Distance		Oriented Segment Hausdorff Distance	
$h(T_p, M_p)$	$h(M_p, T_p)$	$h_{OS}(T_s, M_s)$	$h_{OS}(T_s, M_s)$
115.190 mm	125.156 mm	125.156 mm	125.156 mm

sets of line segments represented by their end points does not do so. That the Hausdorff metric could be used for evaluating the distance of the produced map from the ground truth was hinted at in [132]. It was also used for mobile robot localization [143]. Table 3.5 shows the values of the different measures of disparity between the map produced by the proposed method and the ground truth.

The fact that both the HD and the OSHD are pretty small compared to the dimensions of the environment indicates that the generated map models the ground truth very closely. Also, the fact that $h_{OS}(T_s, M_s) = h_{OS}(M_s, T_s)$ indicates that the segments of the generated map correctly correspond to the segments of the true environment.

3.3 Discussion

Experimental results show that the maps produced by the proposed method are generally better than those produced by two other methods [74,75] reported in the literature in terms of the compactness of the maps and the lengths of the map segments. An important novelty of the proposed algorithm is that, unlike the other algorithms [74, 75] the same setting of the parameters of the proposed algorithm succeeded in building maps of widely differing environments. This testifies to its robustness and efficacy.

The experimental activities were carried out mostly in line with the recommendations made in [132]. This will enable researchers to compare the strengths and weaknesses of our method vis-á-vis other methods [132, 144].

The notion of dimensional error in map, as introduced in this Chapter, is not without its share of shortcomings. It is often not practicable to manually measure the dimensions of interest of all obstacles in an environment. Even if the dimensions are known, associating the map segments to the corresponding planar surface in the real environment can, more often than not, prove to be a non-trivial task. Our second formulation for assessing the accuracy of maps based on HD, and its variant, the OSHD [142] is largely free of these issues. We need not bother about manually measuring the dimensions of the obstacles and figuring out the correspondences of the map segments with the planar surfaces of the obstacles. However, here too, we need to know the locations of the obstacles (or rather, the terminal points of a planar surface) in a given world coordinate system to compute these metrics.

Chapter 4

Monte Carlo localization on line segment-based maps

4.1 Introduction

The capability of localization, which enables a mobile robot to continually estimate its location and heading direction (collectively referred to as *pose*), is a key requirement for autonomous navigation. In some cases, the robot may have absolutely no idea about its pose, except that it is located somewhere in its working environment. More often, it has a fair idea about its pose and all that is required is to keep a continuous track of the pose as it moves around. Inferring the pose of the robot in the former case is referred to as *global localization*, while in the latter case it is referred to as *local localization* or *pose tracking*.

While odometers do provide incremental estimates of pose, such estimates cannot be trusted upon for long periods of time because of integration of measurement errors over the length of travel. Most modern mobile robots are equipped with LRFs, which provide the robot with an accurate range estimate of the nearby objects. Thus, if a spatial model of the environment is available in the form of a map with the robot, it is possible to estimate the pose of the robot.

We have already discussed in Chapter 2 that line segments are a natural choice in the representation of indoor environments and the advantages of such representations over occupancy grid maps.

Monte Carlo Localization (MCL) [76] is a popular approach for mobile robot localization. It employs a Bayes' filter to recursively estimate the posterior probability density function of the robot pose, over the state space, given (noisy) measurements that arrive sequentially in time and probabilistic models of robot motion and perception. It uses a finite set of random samples (also known as *particles*) with associated *importance weights* (or simply, *weights*) to represent, in approximation, the posterior

density. Each particle, representing a robot pose, is a hypothesis as to the true, but unknown pose, while its associated weight provides a relative measure of the "importance" of the sample towards approximation of the target density function. Hereinafter, we shall use the terms sample, particle and hypothesis synonymously.

MCL makes no restrictive assumption about the state space dynamics or the shape of the density function. It is able to track multiple hypotheses about the robot pose and can also trade-off the computational resources available with the accuracy of localization. It has a modular structure and is simple to implement. The benefits associated with the use of line segment-based maps (see page 14) and the effectiveness of MCL as evident, for instance, from [77], are sufficient motivations for implementing MCL on such maps. Unfortunately, studies on the performance of MCL on line segment-based maps have not been extensively reported in the literature. To the best of our knowledge, only [78] has reported it in some detail. A few other authors [145, 146] do talk about MCL on such maps, but they are mentioned in different contexts.

The particle filtering framework, described later in this Chapter, on which MCL is based, only lays down the broad sequence of steps to be followed. The choice of the motion model and the measurement model is left to the programmer. The importance weights, derived from the measurement model, play a key role in the pose estimation process and thus it is imperative to compute the weights in a judicious and efficient manner.

The main theme of this Chapter relates to a proposal on a new, heuristic-driven approach for weight computation on such maps. Apart from being simple to implement and efficient to compute, the novelty in the proposed approach lies in the fact that it just takes into account how closely the actual perception of the robot fits in or is consistent with the environment map when referred to a hypothesized pose. This is in stark contrast to the approach that measures the disparity between the actual perception of robot from its true, but unknown, pose and the predicted perception from the hypothesized pose [78–80]. We shall see in the following Chapter that detailed comparative study reveals our method to be more efficient, robust and accurate than three other competing methods [78–80]. The studies were conducted in three different environments whose maps were built from real data using the method described in Chapter 2.

4.2 Related work

Mobile robot localization has been an area of active research for around three decades. A myriad of localization methods, differing in approach as well as in the intended domain of application has been proposed during these years. In this Section, we briefly review a few approaches to localization before reviewing literature on MCL as applied

to line segment-based maps and on methods of weight computation.

An early but seminal work on localization based on matching raw range scan data with a line segment-based map given *a priori* is due to Cox [128]. Somewhat similar approaches were reported in [147, 148]. Methods that extract features from the range scan data for matching with the map include [149–151]. Another seminal work [152], based on the principle of *iterative dual correspondence*, achieves localization by matching two scans. A method that computes the cross-correlation between two scans to achieve localization was proposed in [153]. A combined scan-matcher using elements of three different methods was proposed in [154]. Localization methods that make use of Hough Transform were reported in [150, 155, 156].

In the later years, localization was formulated as a probability density estimation problem and solved using Bayes' filter [157, 158]. The Kalman filter-based methods postulate the robot pose as a unimodal Gaussian distribution and assume the state-transition model and the measurement model to be linear in its arguments with added Gaussian noise. However, in real-life applications, the Extended Kalman filter (EKF) is often used to address deviation from linearity [143, 155, 156, 159].

Methods based on Markov Localization (ML) maintain a discrete approximation of the probability density function of the pose of the robot. If the state space is represented as a grid array, as in [160], each cell of the grid holds the probability that the robot is currently located in it. The method [161] extends [160] for fast position tracking by maintaining a small search space of possible robot positions centered around the current estimated position. The dynamic Markov localization approach [162] maintains an octree-based hierarchical representation of the state space for dynamically changing the resolution of the discretization based on how certain the robot is about its pose. Experimental evidence suggests [163] that while grid-based ML is more robust, Kalman filter-based methods are more efficient and accurate. To inherit the virtues of both, a hybrid method named Markov-Kalman localization (ML-EKF) was proposed in [164].

MCL utilizes a particle filter-based framework [165] to overcome the computational overhead and memory demands inherent in grid-based ML methods by using a sample-based representation of the probability density [76, 166]. By using two different ways of generating samples, a more robust method that goes by the name of Mixture-MCL was proposed in [167]. A method that improves upon the efficiency of MCL by constraining the path of the robot such that it suffices to sample the pose space only in the vicinity of the path is proposed in [168]. A method for Monte Carlo-based pose tracking is proposed in [169]. To make the method robust against unmodeled objects in the map, an approach called *sampling from a noncorrupted window* is proposed. A hybridization of ML with MCL, wherein the former is used to zero in on a probable region of the state space before the latter is used to precisely estimate the robot pose in that region, is presented in [170]. In [171] the accuracy of MCL-estimated pose at pre-decided

locations was improved by matching scans with previously stored reference scans taken at those locations.

The efficacies of the Bayesian methods have often been augmented by combining them with evolution-based methods [145, 172–174]. The evolutionary methods are formulated as an optimization problem that progressively reduces the initial uncertainty of the robot pose by generating new solutions in the promising regions of the state space by using feedback information available during the evaluation of the solutions. A method for localization based on the Monte Carlo filter augmented with a clustering algorithm and a genetic algorithm is proposed in [145]. The method proposed in [172] combines a genetic algorithm with an EKF for localization. In [173] an adaptive Monte Carlo localization algorithm based on the coevolution mechanism of ecological species is proposed. MCL was combined with Differential Evolution (DE) to achieve a greater accuracy in pose tracking at the cost of increased computation time in [174].

Localization methods based solely on evolution-based metaheuristics have also been proposed. DE-based algorithms for global localization are proposed in [175–178]. A spatially structured genetic algorithm that exploits complex network theory for the deployment of population has been proposed in [179]. The method proposed in [180] is based on the model of species evolution. A localization method based on Harmony Search was proposed in [181]. This method was further improved by hybridization with DE. The suitability of the perturbation vectors in DE-based localization has been studied in [182].

The method due to Espinace et al. [78] deals with MCL on line segment-based maps. It extracts line segments from the current range scan of the robot. This set of segments is called *Observed Segments Group* or *OSG*. Next, the portions of the line segments of the map, as visible from a given sample pose (visible in full or in part) are brought together to form a set of line segments, called the *Map Segments Group* or *MSG*. The method then computes a modified form of the Hausdorff distance [183] between OSG and MSG, followed by a nonlinear transformation of the distance to determine the importance weight associated with the sample pose. A method is presented in [79] for assigning weight to a particle by assessing the Euclidean distance between the laser range scan actually acquired by the robot and the predicted scan obtained from the particle. In the context of presenting an improved resampling technique of a particle filter for tackling the global localization problem, Gasparri et al. [80] proposed a method of weight computation that is similar in concept to the method of [79] but it uses an additional nonlinear transformation to map the computed Euclidean distance values to the corresponding weight values. The method in [78] was improved upon in a recent work [184] by using a different distance measure [142] and using the same nonlinear transformation as used in [80].

Ref. [185] presents a Monte Carlo localization algorithm that works with 3D laser

range scans and a line segment-based map. The 3D point cloud is first reduced to a virtual 2D scan, which is then used to compute the weights of the particles by comparing the predicted scans with the virtual scan. Reference to MCL on line segment-based maps without adequate details may also be found in [145, 146].

4.3 The particle filter framework

In many real-life applications, it is required to estimate the (generally hidden) state of a system using observations, possibly noisy, related to the state. If the observations arrive sequentially in time and the estimation is to be performed online, a recursive solution to the estimation problem is desirable. It obviates the need to store the old observations and to reprocess them when new observations arrive. The Bayesian approach to estimating the state is to construct its probability density function (PDF) given the observations, and to recursively update it, whenever new observations arrive.

Let the observations obtained till time step t be the set of measurements denoted by $\boldsymbol{z}_{1:t} \stackrel{\text{def}}{=} \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t\}$ and let the corresponding states from which these observations are obtained be denoted by $\boldsymbol{x}_{1:t} \stackrel{\text{def}}{=} \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$. Let us assume that both the state $\boldsymbol{x}_t$ and the measurement $\boldsymbol{z}_t$ are conditionally independent of other states and observations, given the immediate preceding state $\boldsymbol{x}_{t-1}$ and the current state $\boldsymbol{x}_t$ respectively.

The PDF of the state at time step t, given the measurement data $\boldsymbol{z}_{1:t}$ and denoted by $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$ is commonly referred to as the *filtering distribution*. It can be obtained recursively from the PDF of the state in the preceding time step, $p(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1})$ using a two stage procedure: *prediction* and *update*.

$$\text{Prediction:} \quad p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1}) d\boldsymbol{x}_{t-1} \tag{4.1}$$

$$\text{Update:} \quad p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t) p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}) \tag{4.2}$$

where η is a normalization constant that ensures that the left-hand-side of equation (4.2) integrates to unity for all $\boldsymbol{x}_t$. The PDF $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ describes how the states stochastically evolve over time and is known as the *state transition probability* while $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ relates the noisy measurements with the state and is known as the *measurement probability*. For cases, where these two probabilities are not linear Gaussian, the PDFs cannot be computed in a closed form. For such cases, we use numerical techniques to compute the PDFs by exploiting the duality that exists between a distribution and the samples drawn from it. Samples originate from a distribution and the latter may be used to approximate the former using techniques like *kernel density estimate* [82].

The problem thus boils down to one of recursively computing at every time step t, the samples that represent $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$ from the samples that represent $p(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{1:t-1})$.

For convenience of formulation, samples are drawn from the joint space $p(\boldsymbol{x}_{1:t}|\boldsymbol{z}_{1:t})$ instead of from the filtering distribution $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$. Finally, the samples corresponding to the time step t are only considered for approximating the filtering distribution at t while those from the previous steps are ignored. The distribution in the joint space may be written as

$$p(\boldsymbol{x}_{1:t}|\boldsymbol{z}_{1:t}) = \eta p(\boldsymbol{z}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{1:t-1}|\boldsymbol{z}_{1:t-1}) \tag{4.3}$$

Since it is generally not possible to sample the *target distribution* $p(\boldsymbol{x}_{1:t}|\boldsymbol{z}_{1:t})$ directly, we take recourse to the method of *Importance Sampling* and draw samples from an appropriate *proposal distribution* $q(\boldsymbol{x}_{1:t}|\boldsymbol{z}_{1:t})$, where each of the distributions are possibly known only up to a normalizing constant. Let the samples at times $1, \ldots, t$ drawn over the joint space be denoted by $\widetilde{\boldsymbol{x}}_{1:t}^{(i)}, \quad i = 1, \ldots, N$. Each of these samples is then assigned an *importance weight* (or simply *weight*) [186] as follows:

$$\widetilde{w}_t^{(i)} \propto \frac{p(\widetilde{\boldsymbol{x}}_{1:t}^{(i)}|\boldsymbol{z}_{1:t})}{q(\widetilde{\boldsymbol{x}}_{1:t}^{(i)}|\boldsymbol{z}_{1:t})}, \qquad i = 1, \ldots, N \tag{4.4}$$

If we choose the proposal distribution in such a way as to factorize as

$$q(\boldsymbol{x}_{1:t}|\boldsymbol{z}_{1:t}) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{z}_t)q(\boldsymbol{x}_{1:t-1}|\boldsymbol{z}_{1:t-1}) \tag{4.5}$$

Equation (4.4) may be rewritten, using equation (4.3) and equation (4.5), as follows:

$$\widetilde{w}_t^{(i)} \propto \frac{p(\boldsymbol{z}_t|\widetilde{\boldsymbol{x}}_t^{(i)})p(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)})p(\widetilde{\boldsymbol{x}}_{1:t-1}^{(i)}|\boldsymbol{z}_{1:t-1})}{q(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)}, \boldsymbol{z}_t)q(\widetilde{\boldsymbol{x}}_{1:t-1}^{(i)}|\boldsymbol{z}_{1:t-1})}$$

$$\propto \frac{p(\boldsymbol{z}_t|\widetilde{\boldsymbol{x}}_t^{(i)})p(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)})}{q(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)}, \boldsymbol{z}_t)} \widetilde{w}_{t-1}^{(i)} \tag{4.6}$$

If we choose $q(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)}, \boldsymbol{z}_t) = p(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)})$, equation (4.6), reduces to

$$\widetilde{w}_t^{(i)} \propto p(\boldsymbol{z}_t|\widetilde{\boldsymbol{x}}_t^{(i)})\, \widetilde{w}_{t-1}^{(i)} \tag{4.7}$$

To ensure that the importance weights add up to unity, we normalize them as

$$w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{k=1}^{N} \widetilde{w}_t^{(k)}} \tag{4.8}$$

where $w_t^{(i)}$ is the normalized weight corresponding to $\widetilde{w}_t^{(i)}$. The weighted samples $\left\{\left(\widetilde{\boldsymbol{x}}_t^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^{N}$ may be used to compute, for instance, the expectation of the distribution $p(\boldsymbol{x}_{1:t}|\boldsymbol{z}_{1:t})$. The above scheme for recursively computing the weights is known as the *Sequential Importance Sampling* (SIS).

Since $\sum_{k=1}^{N} w_t^{(k)} = 1$ and $0 \leq w_t^{(k)} \leq 1$, the weights associated with each sample may be thought of as their corresponding probability masses. Thus, if we draw N samples *with replacement* from the weighted set of samples $\left\{\left(\widetilde{\boldsymbol{x}}_t^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^{N}$ such that probability of drawing a sample is equal to its normalized weight, we get a new set of samples that approximately represent the desired PDF $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$. After resampling, the weights associated with each of the samples are assigned identical values of $\frac{1}{N}$. We then proceed to compute the PDF for the next time step, $p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t+1})$, using the new observation $\boldsymbol{z}_{t+1}$ and the just derived PDF $p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$. The above method for recursive estimation of the posterior density is known as *Sampling Importance Resampling* (SIR) [187, 188].

It is important to note that since $\widetilde{w}_t^{(i)} = \frac{1}{N}$ and the weights are normalized in time step t, the weight update equation as given in equation (4.7) may be written in a simplified form as

$$\widetilde{w}_t^{(i)} = p(\boldsymbol{z}_t|\widetilde{\boldsymbol{x}}_t^{(i)}) \tag{4.9}$$

The samples with associated probability masses are also known as *particles* and since these particles are used to approximate the filtering distribution, the above method is also known as a *Particle Filter*.

In the context of mobile robot localization, the term *belief* is commonly used to refer to the PDF of the robot pose $\boldsymbol{x}_t$ at time t conditioned on all past measurements $\boldsymbol{z}_{1:t}$ and all past controls $\boldsymbol{u}_{1:t}$. Thus, the belief at time t over $\boldsymbol{x}_{1:t}$ is given as

$$Bel(\boldsymbol{x}_t) = p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}) \tag{4.10}$$

where the measurements $\boldsymbol{z}_{1:t}$ pertain to sensory data like laser range scans and the controls $\boldsymbol{u}_{1:t}$ pertain to information related to robot motion like odometry data. For convenience, we shall assume that the perceptual data and the control data arrive in an alternating sequence. Thus, if the belief at time step t before incorporation of the

measurement $\boldsymbol{z}_t$ is required, we shall use the expression

$$\overline{Bel}(\boldsymbol{x}_t) = p(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}) \tag{4.11}$$

The state transition probability in this case takes the form of $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ and is derived from considerations of robot kinematics. Thus, it is also known as the *robot motion model*. The measurement probability $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ depends on the sensing modality used and is also referred to as the *sensor model*. We assume that these models are *stationary* and thus time-invariant.

We start off with a set of samples that represent the initial belief, $Bel(\boldsymbol{x}_0)$, of the robot. This represents our initial knowledge of the robot pose. If we are aware of the initial pose, all the samples are taken concentrated in the corresponding part of the pose space. If, however, we are totally ignorant of the robot pose, we distribute the samples uniformly in the pose space.

Beginning with samples corresponding to $Bel(\boldsymbol{x}_0)$, we first compute samples corresponding to $\overline{Bel}(\boldsymbol{x}_1)$ for $t = 1$ by propagating the initial samples based on the motion model under the action of the control $\boldsymbol{u}_1$. Thereafter, we compute the samples corresponding to $Bel(\boldsymbol{x}_1)$ using the methodology of SIR after incorporating the measurement $\boldsymbol{z}_1$ obtained from the sensor model. The samples representing $Bel(\boldsymbol{x}_1)$ form the initial samples for time step $t = 2$ and the above steps keep repeating for $t = 2, 3, 4, \ldots$.

Since the proposal distribution at time step t is given as

$$\overline{Bel}(\boldsymbol{x}_{1:t}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)\, Bel(\boldsymbol{x}_{1:t-1}) \tag{4.12}$$

the weight computation step reduces to

$$\widetilde{w}_t^{(i)} \propto \frac{Bel(\widetilde{\boldsymbol{x}}_{1:t}^{(i)})}{\overline{Bel}(\widetilde{\boldsymbol{x}}_{1:t}^{(i)})}$$

$$\propto \frac{p(\boldsymbol{z}_t|\widetilde{\boldsymbol{x}}_t^{(i)})\, p(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)}, \boldsymbol{u}_t)}{p(\widetilde{\boldsymbol{x}}_t^{(i)}|\widetilde{\boldsymbol{x}}_{t-1}^{(i)}, \boldsymbol{u}_t)} \frac{Bel(\widetilde{\boldsymbol{x}}_{1:t-1}^{(i)})}{\overline{Bel}(\widetilde{\boldsymbol{x}}_{1:t-1}^{(i)})}$$

$$\propto p(\boldsymbol{z}_t|\widetilde{\boldsymbol{x}}_t^{(i)}) \tag{4.13}$$

Thus, computing the weight of a sample or particle, based on the pose of the particle

and the perceptual data acquired by the robot at any instant t is essential to estimating the PDF of the robot pose at that instant t. In the following Section, we propose a heuristic-driven approach to computing the important weight $\widetilde{w}_t^{(i)}$.

4.4 The proposed method

The MCL method incorporating the proposed approach for importance weight computation assumes that a line segment-based 2D map, M, of the environment is available *a priori*. The map is built in an initial offline phase by merging line segments extracted from registered laser range scans collected from many different scan poses (please see Chapter 2 for details). The map M is a set of $|\,M\,|$ line segments, $M = \{L_1, L_2, \ldots, L_{|M|}\}$, where each segment L_i $(1 \leq i \leq |\,M\,|)$ has a specific orientation, defined by its start point and end point. We further assume that while the robot moves in its environment, it keeps collecting laser range data every half-a-degree in the counterclockwise sense, covering an angular range of 180°, centered on its heading direction. Simultaneously, it also keeps recording the odometry-based estimate of its scan pose. We also assume that the LRF of the robot can reliably sense objects lying within the range $\rho_{max}(= 8000\text{mm})$ and $\rho_{min}(= 50\text{mm})$.

In the remainder of this Section, we first describe the general flow of an MCL method based on the outline given in the preceding Section, and then describe the proposed method of weight computation.

An MCL method (Figure 4.1) starts off with a set of N particles $\left\{\boldsymbol{x}_0^{(i)}\right\}_{i=1}^{N}$ placed on or around the initial pose, assuming the pose to be known (Step 1). This set of particles is then propagated on the basis of the *control action* $\boldsymbol{u}_t$. The control action is estimated using odometry and inflated with artificially added Gaussian noise. In practice, the propagated samples at a given time index are obtained by sampling from an appropriate motion model using the particles in the previous time index and the control action (Step 2). The motion model for sampling that we adopt in our work has been borrowed from Section 5.4 of [158]. In Step 3, the importance weight associated with each propagated particle is computed using the proposed weight computation method. Each of the computed weights is then normalized so that they sum up to unity (Step 4). The pose of the robot is estimated in Step 5 by computing the sum of the particles, where each particle is weighted by its associated (normalized) importance weight. The particles are resampled in Step 6—where the probability of selection of a particle is equal to its importance weight—using the method of *Stochastic Universal Sampling* [189] so that the resulting distribution of particles represents, in approximation, the posterior probability density of the robot pose, given all observations till the current time index [186–188]. This marks the end of one pass in the pose tracking cycle. The time index is subsequently incremented and the cycle is repeated from Step 2

1: Generate particles, $\mathbf{x}_0^{(i)}, i = 1, ..., N$, on or around the known initial pose and set $t \leftarrow 1$

2: For $i = 1, ..., N$, propagate the particles by sampling from the motion model

$$\widetilde{\mathbf{x}}_t^{(i)} = sample\_motion\_model(\mathbf{x}_{t-1}^{(i)}, \mathbf{u}_t)$$

given the control action $\mathbf{u}_t$ with added Gaussian noise.

3: For $i = 1, ..., N$ compute the importance weight $\widetilde{w}_t^{(i)}$ associated with each particle $\widetilde{\mathbf{x}}_t^{(i)}$

4: Normalize the importance weights as

$$w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_{k=1}^{N} \widetilde{w}_t^{(k)}}$$

for $i = 1, ..., N$

5: Compute the pose estimate μ_t as: $\mu_t = \sum_{i=1}^{N} w_t^{(i)} \widetilde{\mathbf{x}}_t^{(i)}$

6: Resample with replacement N particles $\{\mathbf{x}_t^{(i)}\}_{i=1}^{N}$ from the set $\{\widetilde{\mathbf{x}}_t^{(i)}\}_{i=1}^{N}$ based on their importance weights.

7: Set $t \leftarrow t + 1$ and go to Step 2.

Figure 4.1: Monte Carlo Localization algorithm for pose tracking.

onwards (Step 7).

The weight computation procedure, corresponding to Step 3 of Figure 4.1, commences with the extraction of line segments from the laser range scan acquired by the robot from a given scan pose. We use the same procedure for line segment extraction as described in Section 2.4.1. As in Chapter 2, we refer to each line segment extracted from a single range scan as a scan line segment, or a scan segment for short. It is important to note that these segments are all directed, their sense being identical to that in which the LRF obtained the range data from which these segments have been extracted.

In every cycle of pose tracking, all the scan line segments extracted from the same laser range scan are bracketed together first with one particle, then with a second and so on, for all particles one after another. The scan line segments are then transformed to the world coordinate system—in which the particles and the environment map are defined—using the pose of the associated particle with which it has been bracketed. Hereinafter, all references to scan segments shall imply scan segments in the world coordinate system defined with respect to the associated particle. If the particle coincides with the true (but unknown) pose of the robot, all the scan segments broadly match with and overlap on the map segments. Otherwise, the degree of mismatch between the scan segments and the corresponding map segments by and large depends on the extent to which the pose of the particle differs from the true pose. The weight com-

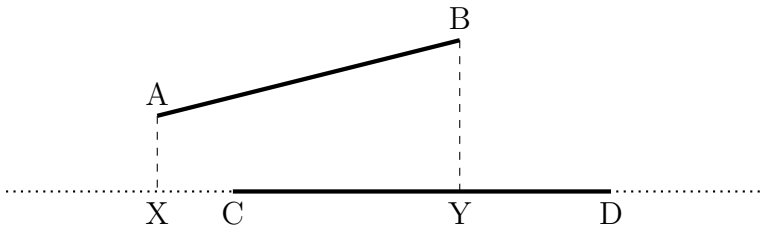


Figure 4.2: The lateral offset and the longitudinal offset of AB from CD provide a measure of their mismatch, where AB represents a scan segment and CD a map segment.

puting method that we propose here is based on the above heuristic; by assessing the extent of mismatch, we estimate the discrepancy between the corresponding particle and the true pose.

The weight computation procedure consists of three steps. In the first step, we establish the correspondence between a scan line segment and a map segment in the sense that both refer to the same physical object in the environment [190, 191]. In the second step, we assess the mismatch between a set of scan segments and the set of corresponding map segments. In the third step, we subject the estimate of mismatch to a nonlinear transformation to arrive at the corresponding value of the importance weight. These weights are then normalized in a subsequent step (Step 4 of Figure 4.1).

In the proposed method, the measure of mismatch, as well as the way to establish the correspondence, rests on the notions of *lateral offset* and of *longitudinal offset* between two directed line segments. The lateral offset, d_{lat}, of segment AB from CD is defined as the arithmetic mean of the (absolute) lengths of the perpendiculars from A and B on the line supporting CD. Referring to Figure 4.2,

$$d_{lat} = \frac{AX + BY}{2} \tag{4.14}$$

where X and Y are the feet of the perpendiculars from A and B respectively on the line supporting CD.

The longitudinal offset, d_{long}, of segment AB from CD is defined as the amount of translation required on the projection XY of AB along the line supporting CD such that XY is completely subsumed in CD. Referring to Figure 4.2, $d_{long}\,(AB, CD) = XC$. If the position and orientation of AB were such that its projection totally overlapped with CD, the longitudinal offset would have been zero. Here, we tacitly assume that the lengths and orientations of AB and CD are such that it is possible to totally subsume XY in CD. We also assume that the points A and C are identical in the sense that both of them are either the start points or the end points of their corresponding segments. Here, it is important to note that these offset measures are not symmetric with respect to their arguments. Even though both are line segments, a map segment

```
1: /* Computes the distance of a given point from the line supporting a given line segment */
2: d(point: p, line segment: l)
3: return (perpendicular distance of p from the line supporting l)
4:
5: /* Computes the lateral distance of a scan line segment from a map segment */
6: d_lat(scan segment: s, map segment: m)
7: return ( average( d(start(s),m), d(end(s),m) ) )
8:
9: /* Computes the longitudinal distance of a scan line segment from a map segment */
10: d_long(scan segment: s, map segment: m)
11: s'=projection of s on the line supporting m
12: if (s' is totally contained in m) then
13:     return (0)
14: else
15:     return (amount of translation on s' required along the line supporting m so
    that s' is totally contained in m)
16: end if
```

Figure 4.3: Pseudo-code for computing the lateral and the longitudinal offsets of a scan line segment from a map segment.

(a representative of ground truth) is intrinsically different from a scan segment (a noisy and partial view of the ground truth). Hence, the arguments to the above measures cannot be interchanged.

Lateral offset implicitly captures the angular deviation as well as the lateral deviation between two segments while longitudinal offset estimates the amount of non-overlap between two segments roughly along their lengths. Their sum gives a heuristic measure of the mismatch between two segments and has no apparent geometrical interpretation.

Let us denote the set of scan line segments associated with a given particle by S and let s_j $(1 \leq j \leq |S|)$ be an individual scan segment such that $s_j \in S$. Similarly, let m_i $(1 \leq i \leq |M|)$ denote an individual map line segment and M the set of all map segments, such that $m_i \in M$. It is to be kept in mind that all s_js and m_is above are directed and have a specific sense of orientation. If $s_p \in S$ and $m_q \in M$ refer to the same physical planar surface of the environment, then s_p will, ideally, not be greater than m_q in length. This is so because in a specific scan only a part of, or at best the entire, planar surface modeled by a map segment will be visible. To identify which scan segment corresponds to which map segment, we first compute the lateral and the longitudinal offsets between a given scan segment and each of the map segments. The pseudo-codes for computing these offsets are given in Figure 4.3. We assume that the function `average(.)` returns the arithmetic mean of the two arguments passed to it,

```
1: AssessMismatch(scan segment set: S, map segment set: M)
2: △ = 0
3: Len = 0
4: flag = FALSE
5: for j = 1, 2, ..., | S | do
6:     D = δ = infinity
7:     for i = 1, 2, ..., | M | do
8:         if ((angle between s_j ∈ S and m_i ∈ M is less than π/6 ) AND (s_j is not
   greater than m_i in length)) then
9:             flag = TRUE
10:            lateral = d_lat(s_j, m_i)
11:            longitudinal = d_long(s_j, m_i)
12:            D = lateral + longitudinal
13:            if D < δ then
14:                δ = D
15:            end if
16:        end if
17:    end for
18:    if (δ < infinity) then
19:        Δ = Δ + δ∗length(s_j)
20:        Len = Len + length(s_j)
21:    end if
22: end for
23: if (flag = FALSE) then
24:    return (-1)
25: end if
26: Δ = Δ/Len
27: return (Δ)
```

Figure 4.4: Pseudo-code for computing the mismatch between a set of scan line segments and the map line segments.

while `start(.)` and `end(.)` return the start point and the end point respectively of the directed line segment passed on to them as argument.

At the heart of the weight computation method lies the function named `AssessMismatch(·)` (Figure 4.4). It is invoked separately for each particle and takes in S and M as its arguments and returns an estimate of mismatch between S and the corresponding elements of M. Thus, the return value of `AssessMismatch(·)` gives us an idea of the extent to which the particle is "away" from the true, but unknown, pose.

For each scan line segment $s_j \in S,\ j = 1, \ldots, |S|$ we determine (i) whether a given map segment $m_i \in M,\ i = 1, \ldots, |M|$ lies within an angle of $\frac{\pi}{6}$ to s_j; and (ii) whether s_j is less than or equal to m_i in length (line 8). If these two conditions are satisfied, m_i is a possible candidate that corresponds to s_j. The specific value of $\frac{\pi}{6}$ in (i) above is empirically chosen and it ensures that if a given m_i differs widely ($\frac{\pi}{6}$ or beyond) from

```
1: ImportanceWeight(mismatch: Δ)
2: if (Δ = −1) then
3:     weight = 0
4: else if (Δ < 1) then
5:     weight = 1
6: else
7:     weight = Δ^−2
8: end if
9: return(weight)
```

Figure 4.5: Pseudo-code for transforming an estimate of mismatch to importance weight.

s_j in orientation, it is not to be considered for possible correspondence with that s_j. To determine the specific m_i (let us call it m_i^*) from among the candidate segments that we believe actually corresponds to s_j, we compute the sum of the lateral offset and the longitudinal offset between s_j and all candidate m_is one after another, and the m_i for which the sum is least is reckoned as the map segment, m_i^* corresponding to s_j. In case the two conditions (in line 8) are not satisfied for a given scan segment, we reckon that no corresponding map segment for it exists and proceed with the next scan segment. If no scan segment from the set S corresponds to any map segment in M, the boolean variable `flag` retains its initial value of *false*. This implies that if the current pose of the robot is supposed identical to that of the particle under consideration, the present perception of the robot is not consistent with the environment. This scenario is signaled by returning a value of -1 (line 24). If the corresponding map segment could be found for at least one scan segment—indicated by the value *true* of `flag` (set in line 9)—the total mismatch between the sets S and M is defined as the weighted average of the offsets of each scan line segment from its corresponding map line segment weighted by the length of the scan line segment (lines 19–20 and 26) and it is this mismatch that is returned in line 27. We assume that `length(·)` returns the length of the segment passed on to it as argument.

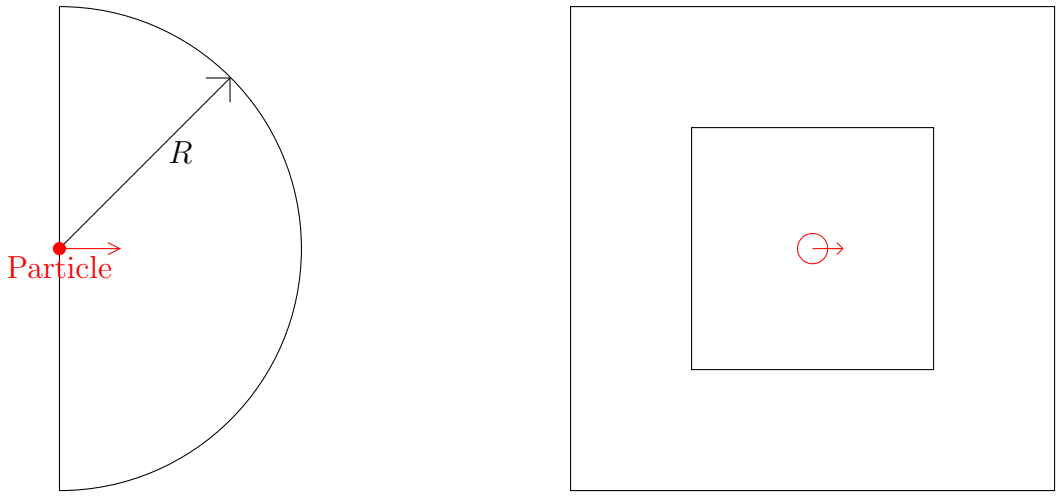
After the mismatch between a set S and M has been estimated, the particle associated with S is assigned weight using the function `ImportanceWeight(·)` that takes in the estimate as its argument (Figure 4.5). The weight corresponding to a mismatch of Δ is given by Δ^{-2} (line 7), provided of course that the mismatch is not -1 or is not less than 1. In the former case, the weight is assigned a value of 0 (line 3) while in the latter case, it is assigned a fixed value of 1 (line 5).

4.5 Speeding up computation

We now propose a strategy which helps in speeding up the above weight computation step. We may sort the map line segments in M according to their lengths in a decreasing order in an initial offline phase. This will obviate the need to compare the length of s_j with every map segment m_i in M (line 8 in Figure 4.4) for the simple reason that for a given j $(1 \leq j \leq |S|)$ and i $(1 \leq i \leq |M|)$, if the length of s_j turns out to be greater than that of m_i, the remaining map line segments m_k where $(i < k \leq |M|)$, will surely be shorter than s_j. The other strategy for speeding up computation stems from the fact that, in general, only a handful of the objects in the environment will be "visible" to a particle at any given instance. Thus, it will suffice to use only the corresponding set of map segments (denoted by, say, M'), instead of the entire set M. In general, those map segments that lie partly or in full within a semicircle of radius $R\,(= \rho_{max})$ and centered on a particle will be visible from the corresponding pose. These line segments thus constitute the set M' for that particle. Here, for simplicity, we ignore the fact that ρ_{min} is different from zero (Figure 4.6a). To avoid computing the set M' for each and every particle in a given time index, we compute the set M'' instead that we believe contains the M's of all the particles in that time index. To determine the set M'', we first make an approximate estimate of the robot pose at t by applying the control action $\boldsymbol{u}_t$ on the robot pose estimated in the preceding time index $t-1$ (in Step 5, Figure 4.1). Thereafter, we consider a square of dimensions $2a \times 2a$ centered on the approximate pose and aligned with its heading direction (Figure 4.6b). We then inflate the square on all sides by an amount R. This results in a larger square of side $2(R+a) \times 2(R+a)$. All map segments that lie partly or in full within the larger square, are considered as belonging to the set M''. We invoke the `AssessMismatch(·)` with the arguments S and M'' respectively, where M'' is a subset of M with its constituent line segments sorted according to their lengths. We need not compute M'' every time we invoke `AssessMismatch(·)` in a given time step as M'' is supposed to include all map segments likely to be seen by all particles in that time step. In determining M'', we make the simplifying assumption that all particles coming from Step 2 of Figure 4.1 in time index t are contained within the square of size $2a \times 2a$. Thus, a should be chosen considering the dynamics of particle propagation. In all the studies we conducted with our method, the value of a was taken as 2000mm.

4.6 Competing methods for weight computation

In this Section, we briefly describe three other methods for weight computation that compete with the proposed method. These methods are separately used to realize Step 3 of the MCL method given in Figure 4.1 and the results of pose estimate compared



(a) (b)

Figure 4.6: (a) All map segments, a part or the entirety of which lies within the semicircle of radius $R(=\rho_{max})$ from a particle forms the set M' for the particle; (b) The smaller square has dimensions $2a \times 2a$, while the bigger square has dimensions $2(R+a) \times 2(R+a)$. The squares are concentric and map segments lying partly or in full within the larger square constitute M''.

with that of our method.

The method of Yaqub and Katupitiya [79] computes the normalized weight of a particle i $(i = 1, 2, \ldots, N)$ as follows:

$$w_i = \frac{1}{N-1}\left[1 - \frac{\sum_{j=1}^{n}\left(\rho_j - R_j^{(i)}\right)^2}{\sum_{k=1}^{N}\left\{\sum_{j=1}^{n}\left(\rho_j - R_j^{(k)}\right)^2\right\}}\right] \tag{4.15}$$

where ρ_j is the actual range sensed by the jth beam of the LRF of the robot during acquisition of the scan, while $R_j^{(k)}$ is the (predicted) range obtained by the kth particle in the corresponding direction. The predicted range value in a given direction from a particle is computed by ray tracing. We assume that there are n range points in a scan and a total of N particles.

In the method of Gasparri et al., [80] the weights are computed as follows:

$$w_i = \frac{\beta}{\sum_{j=1}^{n}\left(\rho_j - R_j^{(i)}\right)^2} \tag{4.16}$$

where β is a normalization constant, whose specific value is unimportant because of subsequent normalization (Step 4 of Figure 4.1).

The method of Espinace et al. [78] first rotates a particle and its associated MSG so that the MSG roughly aligns with OSG (please see page 50) and then computes

the Modified Hausdorff Distance (MHD) [183] between OSG and the rotated MSG for each of the rotated N particles. The weight associated with the ith particle is then computed as

$$w_i = \frac{1 - tanh\left(\frac{2\,((\mathrm{MHD}(\mathrm{MSG}_i\,,\,\mathrm{OSG}))-I_p)}{I_p}\right)}{2} \tag{4.17}$$

where MSG_i is set of map segments likely to be perceived by the ith particle (in entirety or in part), the function $MHD(\cdot)$ computes the MHD between the two arguments passed on to it and I_p is a parameter that regulates how fast the values of the weights decay with MHD. The MSG is computed by ray tracing. In our implementation, we chose the value of I_p in such a way that a particle, which is away from the true pose by half the distance of the particle farthest from the true pose, is assigned a weight of 0.5. With other values e.g., $I_p = 0.2$, the performance of the method was found to deteriorate as we shall see later.

4.7 Discussion

The first step towards computing the importance weight in each of the four methods (Ref. [78–80] and the proposed method) consists in determining the incongruity or mismatch between the actual perception of the robot from its true but unknown pose and the computed perception from the hypothesized pose represented by a particle. In Yaqub and Katupitiya [79] and Gasparri et al. [80], the range data acquired in a given direction is implicitly assumed to correspond to the range data obtained from the map in the same direction using ray tracing. The sum of the squared differences between all such corresponding range data gives the measure of mismatch. The correspondence holds only when the particle lies in the vicinity of the true pose. If the correspondences are incorrect, the measure of mismatch will be erroneous. Espinace et al. [78], on the other hand, does not make any explicit attempt to establish correspondence; it simply takes the difference between two sets of line segments (the OSG and the MSG, with the latter possibly rotated to roughly align with the former) as a measure of mismatch. The proposed method, in contrast to the above methods, includes a procedure for explicitly establishing the correspondence before assessing the mismatch.

The nonlinear transformation (applied on the estimate of the mismatch) governs the extent to which particles are favored in terms of assignment of high weight values vis-à-vis their mismatch values. In Gasparri et al. [80] and the proposed method, weight assigned is the reciprocal of the square of the measure of mismatch. This makes the weight values fall off sharply with increasing mismatch, when the values of mismatch are fairly small. Such an inverse-square relationship exploits, to very high degree,

the knowledge that a particle with a small measure of mismatch is close to the true pose. Since such a particle is assigned a high weight value, it shepherds the remaining particles in the state space and makes a dense cluster in its vicinity, leaving few particles elsewhere. The dense accumulation of particles in the correct region of the state space enables accurate estimation of the robot pose. However, if the measure of mismatch turns out to be substantially in error in the sense that particles that have been assigned small mismatch values are actually far away from the true pose, the particles converge in the wrong region of the state space and the pose estimate becomes erroneous. In an extreme situation, no particle may be present in the region of the state space where the robot is actually present and thus pose tracking may fail completely. The results of experimental and simulation studies reported in the following Chapter also validate this.

Chapter 5

Performance assessment of localization methods

5.1 Introduction

In the preceding Chapter we have had a detailed look at the proposed method of weight computation for Monte Carlo localization on line segment-based maps. We also had a look at three other methods of weight computation from the literature [78–80] which are functionally equivalent to the proposed method. Naturally, motivation arises not only to see how each of these weights computation methods impacts the performances of the localization methods that incorporate them, but also to see how the performances compare against each other.

In this Chapter, we report on the results of localization obtained from simulation [192] and experimental studies. For ease of reference, henceforth, we shall refer to MCL incorporating a specific weight computation method by the first letter of the surname of the first author proposing the method. Thus, MCL–E refers to MCL incorporating the weight computation method proposed in [78]. Similarly, MCL–Y and MCL-G refers to MCL incorporating the methods [79] and [80] respectively. MCL–S refers to MCL incorporating the proposed weight computation method.

The studies were specifically oriented to compare different aspects of the effectiveness and efficiency of the proposed MCL–S method with the three competing methods (MCL–E, MCL–Y and MCL–G) in pose tracking. The source codes that we developed for the four methods were identical, except for the step of weight computation (Step 3 of Figure 4.1 in page 56). The difference in only this step ensured that the variation in average performance among the four methods arose essentially due to the difference in that step. The studies were conducted in the three different indoor environments, hereinafter referred to as Env#1, Env#2 and Env#3. The maps of these environments were built using the method described in Chapter 2 (Ref. [72]) based on the real-life

datasets available in the public domain at Radish [141] and mentioned against each environment given below.

- Env#1: SRI AIC K wing (∼size: 83m×21m);
- Env#2: Department of DIIGA, Engineering University, Ancona (∼size: 47m×47m);
- Env#3: A 375m long corridor of the main building of Chosun University.

5.2 Simulation and experimental results

The simulation and experimental studies that we conduct can be categorized under two heads. We first assess the performances of the methods in real environments using simulated data. In this, we compute and compare the errors in position and in orientation estimate incurred by the fours methods on different trajectories and environments. We also compare their computational overheads by juxtaposing the average time required by each of the methods to compute a single pose estimate. In addition, we investigate the dependence of the error in the pose estimate on the closeness (in terms of translational and angular displacement) between two invocations of the pose tracking method. For the proposed method, we examine the dependence of the error in the pose estimate on the closeness between two invocations of the pose tracking method and the number of particles used for pose estimation. We also see how the instantaneous errors in position and orientation vary over a given trajectory.

In the second category, we compute and compare the errors in position and in orientation estimate incurred by the fours methods on different trajectories in real environments using real data.

5.2.1 Pose estimation using simulated data

The map of a given environment was loaded in a simulator, and a simulated robot driven along a pre-specified path on the map. While the robot moved, laser range data and the corresponding true scan poses (obtained from the simulator) were recorded. The true scan poses were then perturbed by injecting Gaussian noise and the resulting values reckoned as the corresponding odometry estimates of the poses. Hereinafter, all references to odometry-estimated poses in the simulated environment shall imply estimates of the poses obtained in the above way. Thereafter, only the odometry estimates of the scan poses and the range data acquired from those poses were made available to the robot for localization. The poses, as estimated by the localization methods, were then compared with the corresponding true poses for determination of the errors in the pose estimates.

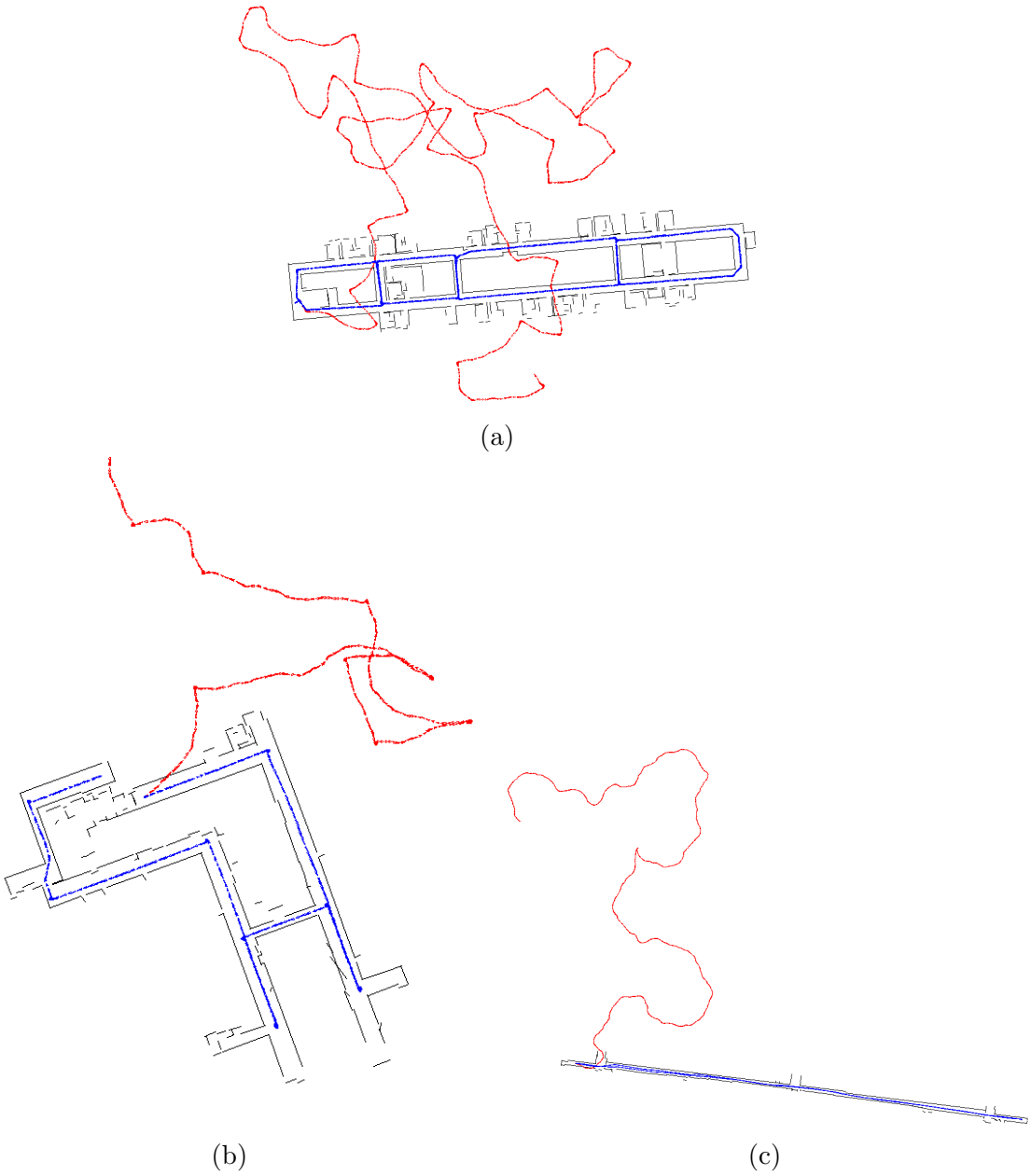
Table 5.1: Three different settings of the noise parameters used in our study

	$\sigma_{\Delta d,d}$ (mm/m)	$\sigma_{\Delta\theta,d}$ (degree/m)	$\sigma_{\Delta\theta,\theta}$ (degree/360°)	$\sigma_{\Delta d,\theta}$ (mm/360°)
Noise #1	10	5	5	2
Noise #2	100	10	10	4
Noise #3	200	20	20	10

For injecting noise, the odometry motion model given in Section 5.4 of [158] was used. The noise parameters (standard deviation of Gaussian distribution) are represented as $\sigma_{\Delta d,d}$, $\sigma_{\Delta\theta,\theta}$, $\sigma_{\Delta\theta,d}$ and $\sigma_{\Delta d,\theta}$ with units mm/m, deg/360°, degree/m and mm/360° respectively. Using a different seed every time, 30 different trajectories were generated for each of the three sets of noise parameters (see Table 5.1 for values), resulting in 90 different trajectories for each environment. The red trajectories in Figure 5.1 are instances of odometry-estimated trajectories obtained by injecting noise (with parameters of set Noise#3) to the corresponding true trajectories, indicated in blue, along which the robot was actually driven in the simulated environment. The total lengths of paths travelled by the robot were around 500m, 165m, and 580m in Env#1, Env#2 and Env#3 respectively.

For evaluation and comparison of their performances, MCL–E, MCL–Y, MCL–G and MCL–S were separately run with 200 particles on each of the different trajectories. A cycle of MCL (Steps 2–7 of Figure 4.1 in page 56) was invoked for pose estimation only when the current pose of the robot differed from the pose on which the previous cycle was invoked by at least 500mm or 5°. The poses here refer to odometry estimates of scan poses. The Euclidean distance between the true position of the robot and the corresponding position as estimated by the localization method under evaluation was computed for each scan pose. In a similar fashion, the angular distance between the true orientation and the estimated orientation was also computed for each scan pose. We refer to these distances as the *position error* and the *orientation error* incurred in the estimation of each scan pose respectively by the localization method under evaluation. When these errors are averaged over all scan poses in a given run along the entire trajectory, we refer to the resulting quantities as the *average position error* and the *average orientation error* respectively for the trajectory incurred by the corresponding localization method. Since these average errors pertain to a single run of a localization method on a particular trajectory, they are not suitable figures-of-merit on which to compare the methods with. So, we consider the mean of the average errors of 30 runs, where each run is taken on a different trajectory that is generated from the same set of noise parameters of Table 5.1 using a different seed. We also consider the 95% confidence interval of the mean computed using a t-distribution.

In Figure 5.2, we show how the mean of the average errors in position and in



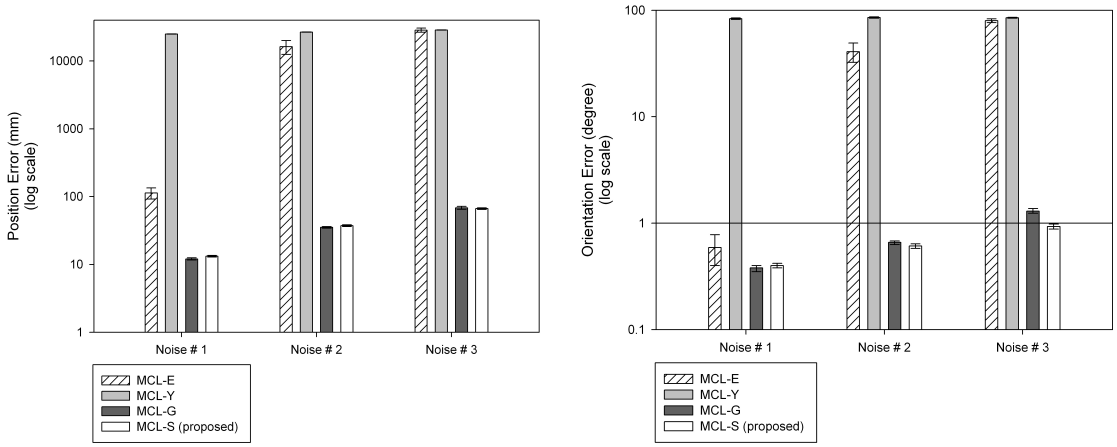
(a)

(b) (c)

Figure 5.1: The blue trail shows the path along which the robot was actually driven, while the trajectory reconstructed by adding Gaussian noise with parameters corresponding to Noise#3 is shown in red in (a) Env#1; (b) Env#2; and (c) Env#3.

orientation, as estimated by the four methods, compare against each other for different levels of injected noise in Env#1. It is important to point out that on quite a few occasions MCL–G failed to estimate the poses after a while (we discuss the reasons for failure in Section 5.3). To be specific, MCL–G failed on eight occasions out of 30 attempts for Noise#3. Thus, the mean of the average position and of the orientation errors and the confidence intervals were based on $22(= 30 - 8)$ runs for MCL–G for Noise#3.

From Figure 5.2, it is evident that the proposed MCL–S and the MCL–G easily



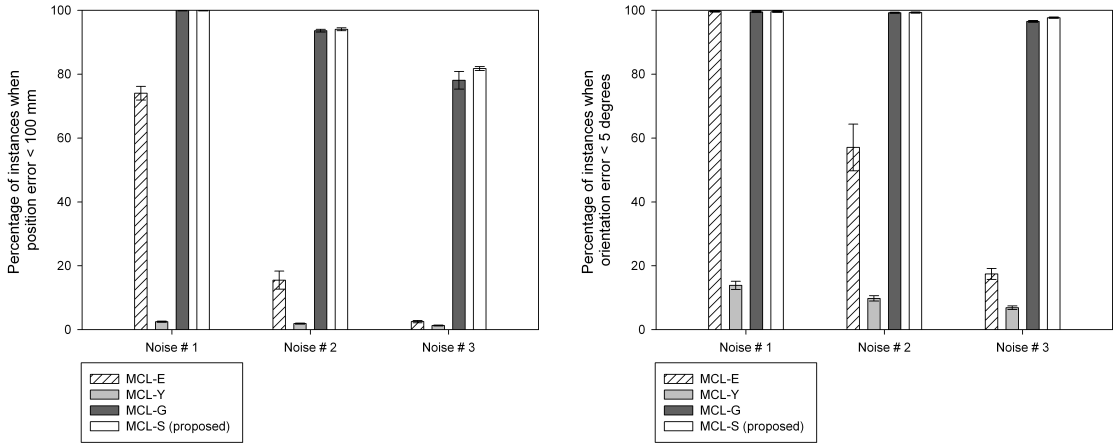
(a) (b)

Figure 5.2: Comparison of the accuracy of pose estimation for different levels of injected noise in Env#1: (a) the mean of the average position errors; and (b) the mean of the average orientation errors. Both the errors are plotted in log scale. The error bars indicate 95% confidence interval. MCL-G has failed to track poses on 8 occasions out of 30 for Noise#3.

outperform the other two methods insofar as accuracy of pose tracking is concerned. These two methods are roughly at par, even though MCL–S makes a slightly better estimate of the orientation for Noise#3. Though accurate, MCL–G is not adequately robust as it lost track of the robot on eight occasions out of 30 attempts for Noise#3. As stated in page 63 (Section 4.6), the above results (as also all other results reported in this Chapter) for MCL–E were obtained with $I_p = 0.5$. For lower values of I_p, e.g., $I_p = 0.2$, the estimates improved for Noise#1, but deteriorated for Noise#2 and Noise#3 and MCL–E even failed to track poses on a couple of occasions.

In Figure 5.3, we show the mean of the number of occasions, expressed as a percentage, when the position errors and the orientation errors in the estimated scan poses were less than 100mm and less than 5° respectively as the robot traversed its trajectory in Env#1. As in Figure 5.2, for each set of noise parameters, every method was run on 30 different trajectories and only those many runs were used in computing the final results as could be successfully completed. The number of scan poses in each trajectory ranged between 1200 and 1550.

From Figure 5.3 it is evident that in almost all cases the estimates of the poses differ from the corresponding true poses by less than 100mm and 5° for MCL–G and the proposed MCL–S for Noise#1. With an increase in the level of injected noise, the number of occasions when the differences are so, drop. But even at Noise#3, the number of such occasions was around 80% (for position estimate) and 96% (for orientation estimate) for MCL–G and MCL–S, though MCL–S was marginally better in both estimates. For the proposed MCL–S, we never encountered a situation when



(a) (b)

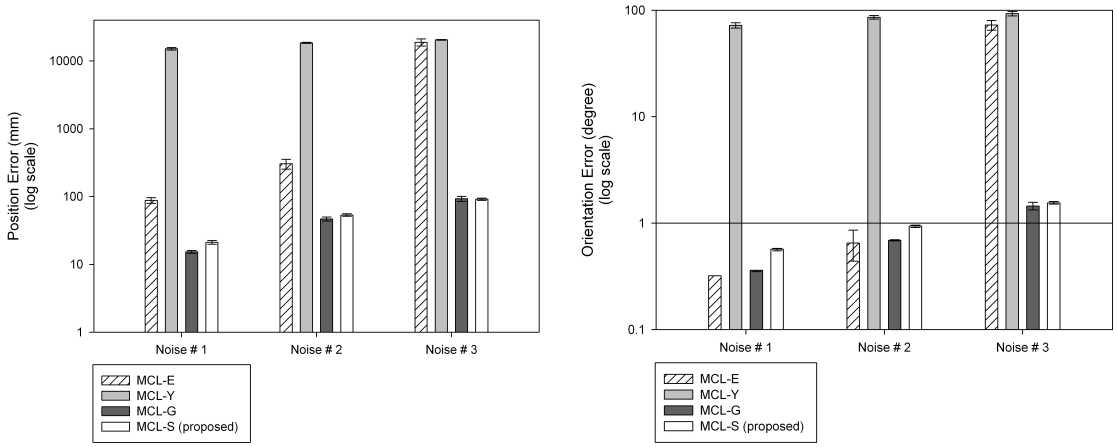
Figure 5.3: Comparison of the percentage of occasions when (a) the estimated positions differ from the corresponding true positions by less than 100mm; (b) the estimated orientations differ from the corresponding true orientations by less than 5°. The error bars represent 95% confidence intervals. The results pertain to Env#1 where MCL–G failed on 8 occasions out of 30 for Noise#3.

the estimated position differed from the true position by more than 1m. But, there were a few such occasions for MCL–G with Noise#3. For Noise#1, MCL–E estimated the orientations with less than 5° error in almost all cases. But, it performed poorly for Noise#2 and Noise#3. Irrespective of the levels of injected noise, the performances of MCL–Y were poor. The number of occasions when the position error was more than 1m was over 90% for MCL–Y for all the three injected noise levels. Thus Figure 5.3 serves to attest to the accuracy of both MCL–G and MCL–S from another standpoint.

As in Env#1, the four methods were also run in Env#2 and Env#3 for different levels of injected noise. The results for Env#2 are graphed in Figures 5.4 and 5.5 while those for Env#3 are graphed in Figures 5.6 and 5.7.

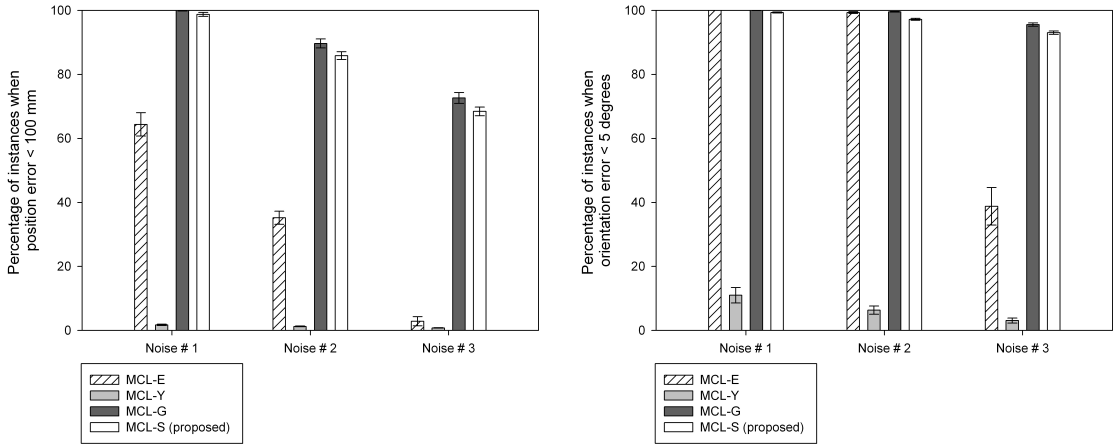
Scrutiny of Figures 5.4 through 5.7 reveals that the relative performances of the methods in Env#2 and Env#3 are fairly similar to that in Env#1. MCL–Y performed poorly all through. MCL–E performed poorly in Env#3 but was relatively better off in Env#2. MCL–G was fragile, as usual, when odometry noise was moderate and high, but MCL–S was robust and accurate all through. It is important to note that MCL–G failed on all 30 occasions for Noise#3 in Env#3.

For convenience of comparison, in Figure 5.8 we plot the percentage of runs (out of 30) in which MCL–G and MCL–S could track the robot pose for the entire trajectory in different environments and for different levels of injected noise. From the Figure it is evident that while MCL–S could track the trajectories in all cases, MCL–G succeeded in tracking complete trajectories in only a few cases. For Noise#3 in Env#3 it could not completely track even a single trajectory. Thus, MCL–S has clearly outperformed



(a) (b)

Figure 5.4: Comparison of the accuracy of pose estimation for different levels of injected noise in Env#2: (a) the mean of the average position errors; and (b) the mean of the average orientation errors. Both the errors are plotted in log scale. The error bars indicate 95% confidence interval. MCL–G has failed to track poses on 12 occasions out of 30 for Noise#3.

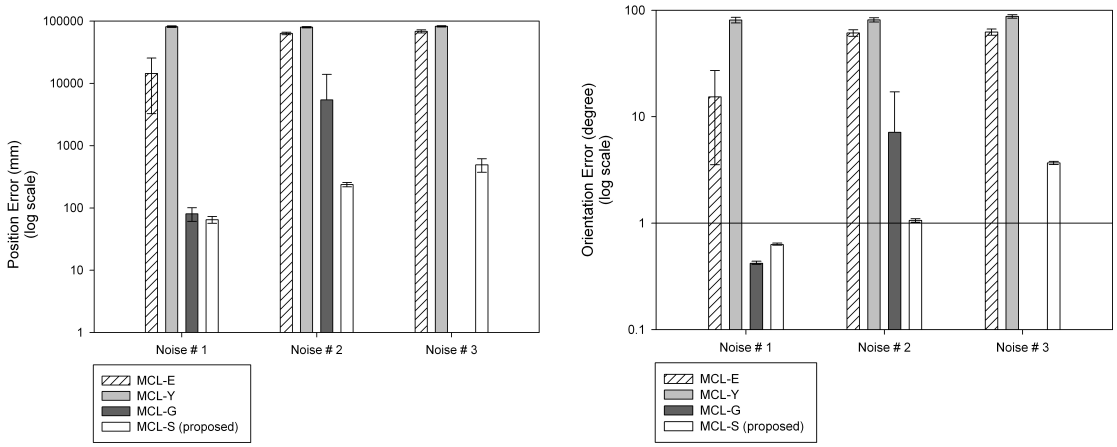


(a) (b)

Figure 5.5: Comparison of the percentage of occasions when (a) the estimated positions differ from the corresponding true positions by less than 100mm; (b) the estimated orientations differ from the corresponding true orientations by less than 5°. The error bars represent 95% confidence intervals. The results pertain to Env#2 where MCL–G failed on 12 occasions out of 30 for Noise#3.

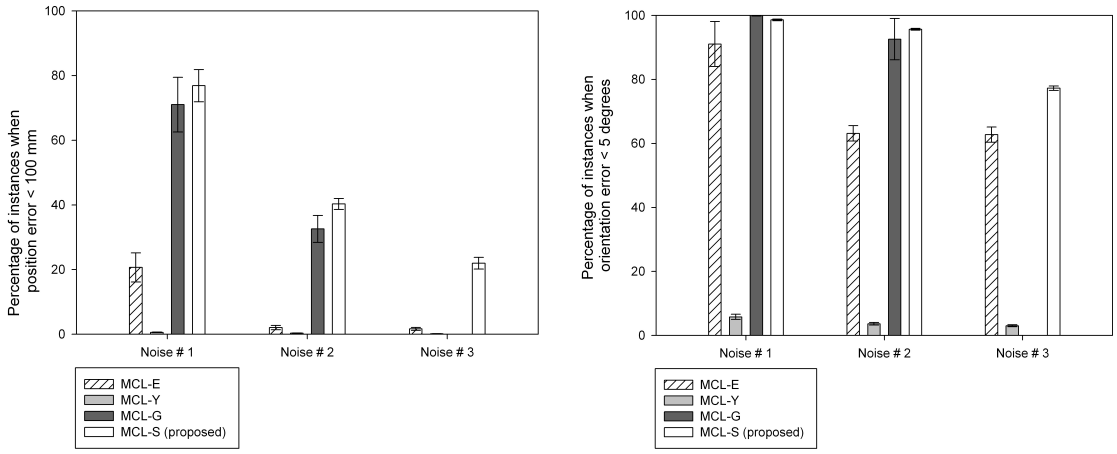
MCL–G insofar as robustness is concerned.

It may be noted that the performances of the methods are worse in Env#3 compared to their performances in Env#1 and Env#2. This is attributable to the fact that Env#3 is a long corridor. As the robot moves down the corridor, it mainly sees parallel walls. The robot can sense and ascertain its distance from these walls but is rather uncertain



(a) (b)

Figure 5.6: Comparison of the accuracy of pose estimation for different levels of injected noise in Env#3: (a) the mean of the average position errors; and (b) the mean of the average orientation errors. Both the errors are plotted in log scale. The error bars indicate 95% confidence interval. MCL-G has failed to track poses on 4 occasions out of 30 for Noise#2 and all 30 occasions for Noise#3. Hence, the bars corresponding to MCL–G for Noise#3 is missing.



(a) (b)

Figure 5.7: Comparison of the percentage of occasions when (a) the estimated positions differ from the corresponding true positions by less than 100mm; (b) the estimated orientations differ from the corresponding true orientations by less than 5°. The error bars represent 95% confidence intervals. The results pertain to Env#3 where MCL–G failed to track poses on 4 occasions out of 30 for Noise#2 and all 30 occasions for Noise#3. Hence, the bars corresponding to MCL–G for Noise#3 is missing.

about its position along the length of the corridor for most part of its travel. This leads to relatively poor localization in Env#3.

In Figure 5.9 we juxtapose the computation time taken by the four methods to estimate a single scan pose (i.e., to perform steps 2–7 of Figure 4.1 in page 56) using

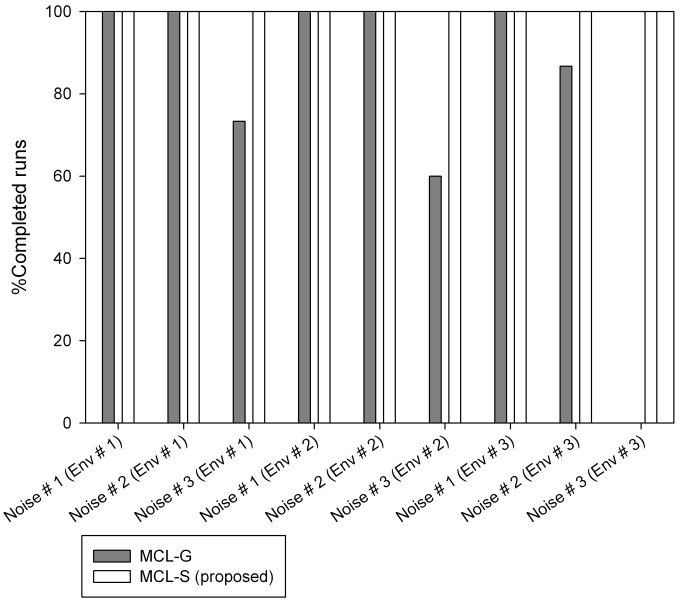


Figure 5.8: Comparison of the percentage of runs (out of 30) in which MCL–G and MCL–S could complete tracking the robot pose using simulated scan data for the entire trajectory.

200 particles. The methods were implemented in C++ and run on a PC with 2.8 GHz Intel Core 2 Duo processor and 4 GB RAM running Debian 5.0 Linux operating system. This Figure reveals that the proposed MCL–S method is computationally more efficient (by about an order of magnitude) than the other three methods. The efficiency can be attributed to the fact that, unlike the other methods, the proposed method does not perform any ray tracing, which is a very compute-intensive process. In our implementation, we invoke `AssessMismatch(·)` with M'' instead of M as the second argument (please see Section 4.5). By doing so, we succeeded in bringing down the computation time by about 30%. Using a somewhat similar approach for the other three methods, we have succeeded in bringing down their computation times as well by over 80% without compromising on their performances. The differences in computation time for the other three methods across different environments arise essentially due to the difference in the numbers of line segments in their corresponding maps. Because of a smaller computational overhead, MCL–S permits more frequent pose updates without overburdening the onboard processor. Frequent pose updates abate inflow of large odometry errors and thus help improve the accuracy of pose estimates.

We have already noted earlier that we choose the interval between two invocations of the pose estimation cycle for updating the robot pose in such a way that the robot has moved by at least 500mm or 5°. This interval (hereinafter called *update interval*) has important ramifications. If the chosen interval is large, more odometry error will creep in and that will make accurate pose estimation more challenging. If this interval

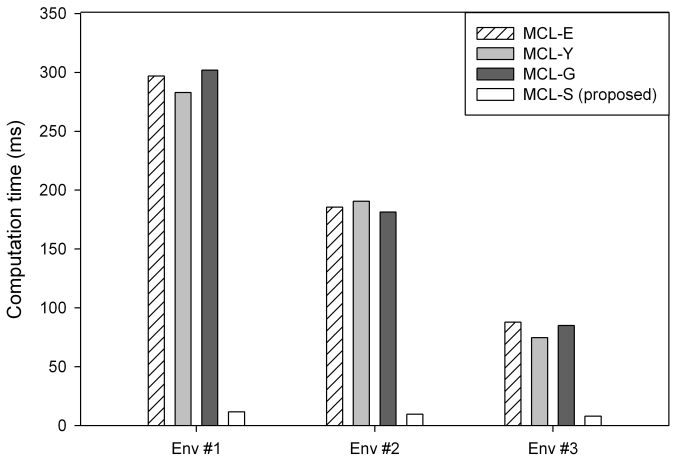
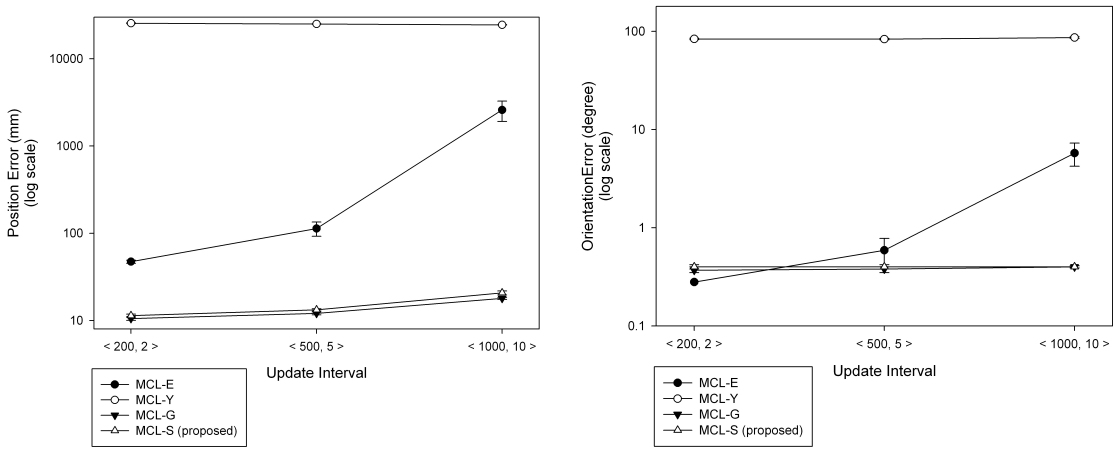


Figure 5.9: Comparison of the computation times (in ms) required to estimate a single pose by the four methods using 200 particles in three different environments.

is too small, more frequent invocations of the cycle will be necessary. This will no doubt make accurate pose estimation simpler, but will leave precious little time for the CPU onboard the robot to perform other important tasks like path planning and trajectory tracking. Thus, a desirable attribute of a good localization method is that it should continue making accurate pose estimates even in the face of increased update intervals. To see how the four methods perform when confronted with different update intervals, in Figure 5.10 we graph the variations in their position estimation error and orientation estimation error (mean of average position and average orientation error on 30 different trajectories) with three different update intervals using 200 particles and with injected noise corresponding to Noise#1 for Env#1. The graphs show that as the update interval increases, so does the error in position and in orientation estimate. This is natural because with an increase in the distance, translational or angular, travelled by the robot, more error gets introduced into its odometry estimate making it more uncertain about its whereabouts. From the graphs, it is obvious that the performances of MCL–G and MCL–S are much better than MCL–Y and MCL–E for all the three update intervals. The performance of MCL–E deteriorates drastically with increasing update interval. The performances of MCL–G and the MCL–S are roughly at par, but while the proposed MCL–S could successfully track poses for the entire trajectory with an update interval of 1000mm or 10° for all the three sets of noise parameters on all 30 occasions, MCL–G failed to keep track of the robot on 28 out of 30 occasions (for Noise#2) and on all the 30 occasions (for Noise#3). This clearly brings out the fragile nature of MCL-G in trying circumstances!

Now that we have compared our method with the three others, we take a closer look into our method. First, continuing with the study on the dependence of the posi-

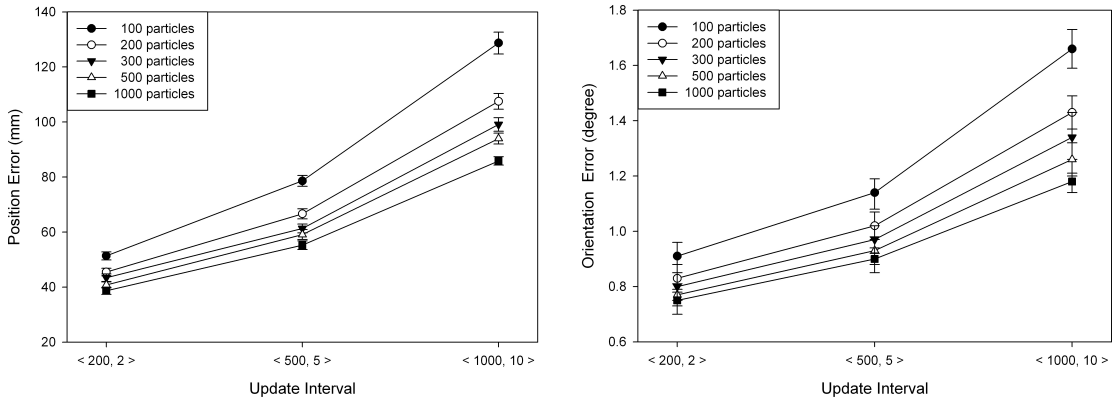


(a) (b)

Figure 5.10: (a) Mean of the average position error (in mm) in log scale versus update interval; and (b) Mean of the average orientation error (in degree) in log scale versus update interval, for Noise#1 with 200 particles in Env#1. The update intervals are denoted by a 2-tuple like $< 500, 5 >$ with units mm and degree respectively. The update intervals used are $< 200, 2 >$, $< 500, 5 >$ and $< 1000, 10 >$. The error bars indicate 95% confidence interval.

tion estimates on update intervals, we investigate into its performance using different numbers of particles. Figure 5.11 shows that though a wider update interval introduces more error into the pose estimation process, an increased number of particles can offset the resulting increased uncertainty to some extent by sampling the state space more densely. For a fixed update interval and number of particles, the error in position estimate increases as the injected noise in the odometry estimate increases because it again makes the robot more uncertain about its pose. The Figure corroborates the robustness of the proposed MCL–S by making it abundantly clear that MCL–S can continue to track the robot with fairly good accuracy even in the face of very high noise (viz., Noise#3) and large update interval (1000mm or 10°). Though this Figure pertains to Env#1 for Noise#3, similar graphs were obtained for Noise#1 and Noise#2, but the magnitudes of the errors were correspondingly smaller in those cases.

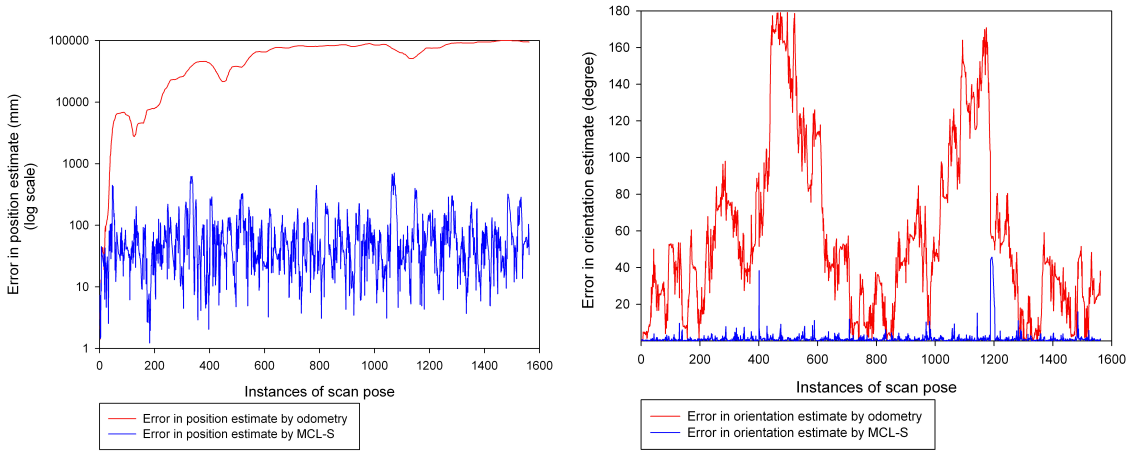
In Figure 5.12, we show how the error in the estimate of each scan pose incurred by the proposed MCL–S method varies over the entire path of travel for one particular instance of the run on a trajectory generated with Noise#3 in Env#1. In Figure 5.12a, the position errors have been plotted on log scale. While the graph in red shows the error in the odometry estimate of the position, the one in blue shows the position error incurred by MCL–S. Whereas the error in odometry grows without bound with the length of travel and becomes as high as 100m, the error incurred by MCL–S remains mostly within 100mm and between 100mm and 1m in a few cases. In Figure 5.12b, the error in the orientation estimate by odometry is shown in red while that by MCL–S is



(a) (b)

Figure 5.11: Mean of the (a) average position error (in mm) versus update interval; and (b) average orientation error (in degree) versus update interval, for Noise#3 in Env#1. The update intervals are denoted by a 2-tuple like $< 500, 5 >$ with units mm and degree respectively. The numbers of particles used are 100, 200, 300, 500 and 1000 and the update intervals are $< 200, 2 >$, $< 500, 5 >$ and $< 1000, 10 >$. The error bars indicate 95% confidence interval.

shown in blue. It is important to note that the orientation error cannot grow beyond 180°. The orientation error of MCL–S is largely within 1° and only on a few occasions it goes as high as 10° or beyond.



(a) (b)

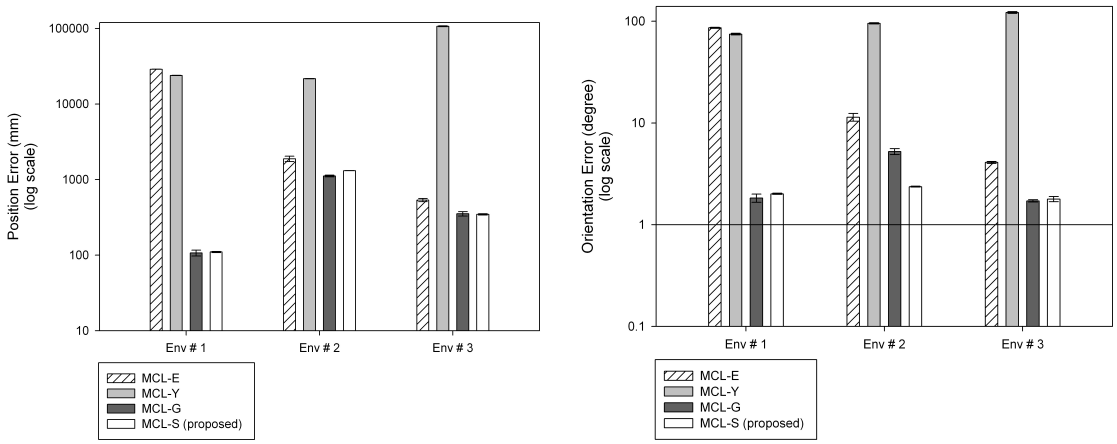
Figure 5.12: (a) Position error (in mm) (in log scale); and (b) Orientation error (in degree), versus pose instances along the length of travel incurred by MCL–S.

5.2.2 Pose estimation using real data

In this study, we used the same maps viz., Env#1, Env#2 and Env#3 but instead of driving a simulated robot through it to generate data, we utilized the real data

(consisting of odometry estimated scan poses and laser range scans) collected from the real environment while a real robot moved along a specified trajectory of approximate length 286m, 208m and 327m in Env#1, Env#2 and Env#3 respectively. Since, in a real environment the true values of the scan poses are not directly available, we used the estimates of the scan poses obtained through scan matching as reasonable estimates of the corresponding true poses.

In this comparative study, we invoked each of the four methods using 200 particles and update interval of $< 250, 10 >$ i.e., 250mm or 10°. The mean of the average position and average orientation error of 30 runs incurred by the four methods are shown in Figure 5.13. Even in the case of real data, the average from multiple runs were taken to study the dependence of the methods on the distribution of particles since the distribution was done with randomly injected noise in each cycle of pose tracking (step 2 of Figure 4.1).



(a) (b)

Figure 5.13: Comparison of the accuracy of pose estimation using real data (a) the mean of the average position error; and (b) the mean of the average orientation error. Both the errors are plotted in log scale. The error bars indicate 95% confidence interval. MCL–G failed to track the pose for the entire trajectory on 23 out of 30 occasions in Env#1. In Env#2, MCL–G failed on 10 out of 30 runs, while MCL–E failed on 3 out of 30 runs. Only that many runs were used in computing the mean errors as could be successfully completed.

Figure 5.13 reveals that both MCL–G and MCL–S outperform the other two methods insofar as accuracy of pose tracking is concerned. But MCL–G is too fragile as it failed to track the pose along the entire trajectory on 23 out of 30 occasions in Env#1 and on 10 out of 30 occasions in Env#2. (For a graphical comparison, please see Figure 5.14). MCL–E failed too, but only on 3 occasions out of 30 in Env#2. While taking a note on the magnitudes of the mean errors, it must be kept in mind that here, unlike in the previous study using simulated data, the true poses of the robot

are not accurately known. For instance, many of the scan poses estimated through scan matching (and reckoned as the true poses) did not appear plausible in Env#2 as they were seen to be outside the map when plotted. Moreover, the robot was imparted backward motion on some part of the trajectory in Env#2. These are attributable to the relatively high error in pose estimate in Env#2. Moreover, in a real environment, the map error (i.e., how closely the map models the actual environment) also introduces error in the pose estimation.

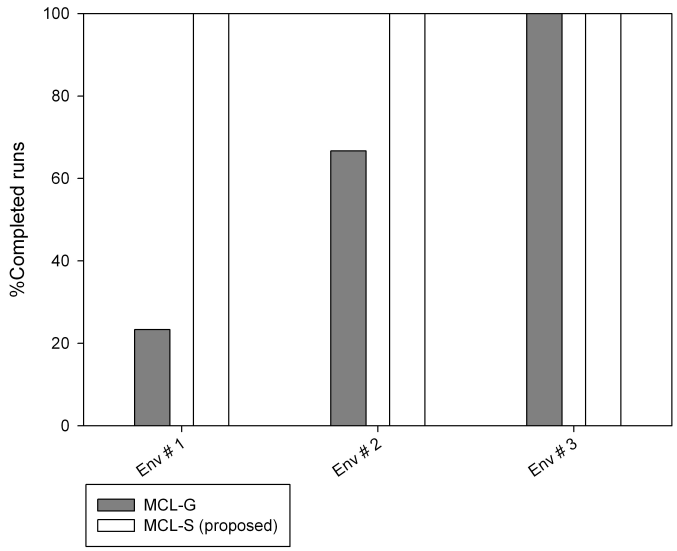
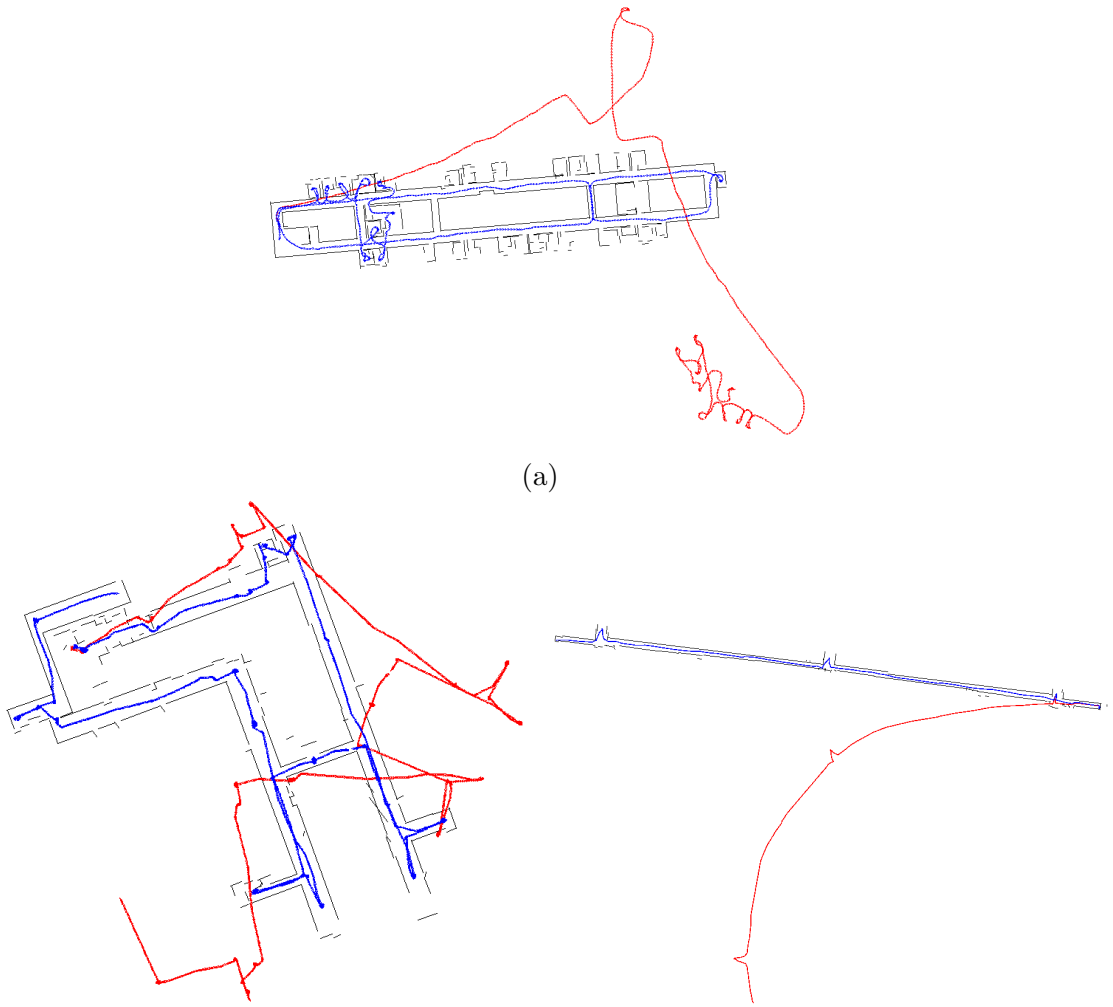


Figure 5.14: Comparison of the percentage of runs (out of 30) in which MCL–G and MCL–S could complete tracking the robot pose using real scan data for the entire trajectory.

In Figure 5.15, we show the trajectory of the robot, estimated using odometry data (in red) and the corresponding trajectory estimated by the proposed MCL–S method (in blue). The odometry-estimated trajectory is too wayward and has moved outside the map, while the trajectory estimated using MCL–S is generally nicely confined inside the map. This study once again shows the accuracy and robustness of the proposed MCL–S method.

5.3 Discussion

We have already seen in the preceding Chapter that the method for computation of the importance weights consists in first making an estimate as to how distant a particle is from the true, but unknown, pose of the robot. This estimate is then nonlinearly transformed so that a particle which is supposedly close to the true pose appears closer than it really is compared to another particle which is not so close. This transformation facilitates making a dense cluster of samples in the supposed neighborhood of the robot



(a)

(b) (c)

Figure 5.15: The trajectory estimated using real odometry data is shown in red, while the trajectory estimated by the proposed MCL–S method using real odometry and range data is shown in blue in (a) Env#1; (b) Env#2; and (c) Env#3.

pose by depriving other regions of them. A dense collection of samples in a small region of the state space enables accurate estimation of the robot pose provided of course the robot is actually present in the densely sampled region. In case a particle that is actually far removed from the true pose, is erroneously estimated to be close to the true pose, and the nonlinear transformation scales up its closeness in a very pronounced way, almost all the samples would be drawn into an incorrect region of the state space eventually leading to a failure in tracking the robot. A nonlinear transformation that does not scale up the closeness estimate in such a pronounced way could possibly save the day in such a situation by allowing samples to be present in other regions of the state space for tracking of the robot in those regions. But such a transformation will also not promote formation of a dense cluster of samples thereby impairing accurate pose estimation. The choice of the transformation function, thus, regulates the tradeoff between accuracy and robustness. We can, however, gain in

accuracy without compromising much on robustness, if we can devise a method for reliable estimation of the closeness of a particle to the true robot pose.

Since the procedure employed by MCL–G for assessing the mismatch depends on the correctness of the correspondences, which is not very robust, and at the same time uses an inverse-square transformation, the procedure is prone to failures. As seen in Section 5.2, this is the reason behind the occasional failures of MCL–G though it otherwise gave accurate pose estimates. MCL–Y employs an identical procedure for mismatch assessment as MCL–G, but it does not use any transformation of the mismatch values. It simply normalizes the mismatch values and reckons these values as the corresponding weights. (Strictly speaking, therefore, step 4 of Figure 4.1 is superfluous for this method). Since this method does not exploit, to an appreciable extent, the information of the particles with high weights, particles do not densely cluster around the region of the state space where the robot is possibly situated after the resampling step. Thus, this method fails to give an accurate estimate of the pose. For the same reason, it never fails as a few particles always remain scattered over the map. The nonlinear transformation that MCL–E uses, does exploit the information associated with particles with high weights, but not to the extent (which is regulated by I_p) that MCL–G or the proposed method does. Moreover, its methodology for mismatch assessment is also not very accurate. Thus, so far as pose tracking is concerned, it is less accurate than both MCL–G and MCL–S, but somewhat more robust than MCL–G. However, if a small value of I_p like $I_p = 0.2$ is used, the performance of MCL–E improves for low noise and small update interval, but deteriorates substantially for high noise and large update intervals. Moreover, if MCL–E is modified so as to use the same nonlinear transformation that MCL–G and the proposed method use, its accuracy in pose tracking improves but it becomes more fragile.

The above discussion, considered in the backdrop of the simulation and the experimental results, makes it abundantly clear that the accuracy and robustness of a pose tracking method depend both on the method adopted to assess the mismatch and the nonlinear transformation used. Since our method uses a transformation function that falls off sharply with the extent of mismatch and is buttressed by a robust and accurate method of assessing the mismatch, we have been able to achieve accurate and robust pose tracking all through.

Chapter 6

Summary, conclusions and future directions

6.1 Introduction

We recall that we had started off with the objective of achieving robust and accurate localization of mobile robots equipped with LRFs in indoor environments that remained practically unchanged for extended periods of time. Accordingly, we had set out to devise

- an offline method for building maps of indoor environments by merging line segments extracted from registered laser range scans [72]; and

- a robust and accurate method for localization on such maps based on the Monte Carlo framework [73].

Our objective was guided by the fact that in many real-life applications, a robot was required to navigate continuously in environments that remained practically invariant over long periods of time. For all those applications, it was more appropriate to build a map of the working environment first and then use it for localization, rather than operate the robot in an online SLAM mode, which was complex, computation-intensive and less accurate. By decoupling mapping from localization, the robot could be absolved of the job of map building while it was performing its assigned duty.

In the following Section, we summarize the contribution of this thesis towards achieving the above objectives and the conclusions arrived at. We segregate our comments under the two heads of map building and localization. In the subsequent and concluding Section of this thesis, we discuss on the avenues for future work that emanate from the present research.

6.2 Summary and conclusions

6.2.1 Line segment-based map building

The recent trend of deploying mobile robots in large indoor environments has necessitated the development of highly scalable mapping methods. The occupancy grid representation puts a heavy, often infeasible, demand on the computer memory and does not scale well with the environment size. Among other virtues, line-segments based maps scale well with the environment size.

In this thesis, we have proposed an offline method for building global maps of indoor environments based on line segments using laser range data. Central to this method is a new formulation for identifying, and then merging into one, all line segments that represent the same planar surface in the environment. The method has been successful in accurately building maps of large environments from real data available in the public domain as well as from simulated data. It has also succeeded in building a map of a medium-sized room accurately. Experimental results show that the maps produced by the proposed method are generally better than those produced by two other methods reported in the literature in terms of the compactness of the maps and the lengths of the map segments. An important novelty of the proposed algorithm is that, unlike the other algorithms, the same setting of the parameters of the proposed algorithm succeeded in building maps of widely differing environments. This testifies to its robustness and efficacy. Maps built by this method could be used in such mobile robot applications where the operational environment of the robot remains practically invariant over long periods of time. In these applications, mapping can be decoupled from localization, and the robot can be absolved of the job of map building while it is performing its assigned duty. The experimental activities were carried out mostly in line with the recommendations made in [132]. This will enable researchers to compare the strengths and weaknesses of our method vis-à-vis other methods [132, 144]. We also advocated the use of directed line segments in the representation of maps when the range data have been acquired using LRF.

We have also proposed ways of assessing the goodness of maps. Making such assessments is tricky, as the basis on which such assessments can be made is often application-specific. Moreover, the unavailability of ground truth data makes an objective assessment of map accuracy extremely difficult. The shortcomings notwithstanding, our formulations for assessment of the goodness of maps deserve attention as they may be considered as initial results in map quality assessment over which more elegant measures may be devised by researchers in future as such measures are mostly unavailable.

6.2.2 Monte Carlo localization on line segment-based maps

MCL is a preferred approach in localization as it makes no prior assumption on the state space dynamics or the shape of the probability density function associated with the random variable modeling the robot pose and can easily tradeoff the computational resources with the accuracy of localization. Line segment-based maps provide a very scalable and compact representation of indoor environments. Thus, motivation naturally arises to implement and study the performance of MCL on line segment-based maps. Unfortunately, there is precious little literature on the use of MCL on segment-based maps.

The particle filtering framework, on which MCL is based, keeps open before the programmer the choice of strategy to be adopted for computing the importance weights associated with the particles. And it is this strategy that, generally speaking, endows a particular implementation of MCL with attributes like robustness, accuracy and efficiency or robs it of them. In this thesis, we have proposed a heuristic-based method for computing the importance weights of an MCL in the context of pose tracking on line segment-based maps and have, through extensive studies using both simulated and real data, corroborated that the method is both accurate and robust in addition to being computationally efficient. Furthermore, the proposed method proved to be superior to three other methods of weight computation [78–80] insofar as endowing MCL with all the above attributes are concerned. For achieveing accurate localization, it was not required for the environment to be modified in any particular way, for instance, by mounting laser reflectors at vantage points.

The method for computation of the importance weights consists in first making an estimate as to how distant a particle is from the true, but unknown, pose of the robot. This estimate is then nonlinearly transformed so that a particle which is supposedly close to the true pose appears closer than it really is compared to another particle which is not so close. This transformation facilitates making a dense cluster of samples in the supposed neighborhood of the robot pose by depriving other regions of them. A dense collection of samples in a small region of the state space enables accurate estimation of the robot pose provided of course that the robot is actually present in the densely sampled region. In case a particle that is actually far removed from the true pose, is erroneously estimated to be close to the true pose, and the nonlinear transformation scales up its closeness in a very pronounced way, almost all the samples would be drawn into an incorrect region of the state space eventually leading to a failure in tracking the robot. A nonlinear transformation that does not scale up the closeness estimate in such a pronounced way could possibly save the day in such a situation by allowing samples to be present in other regions of the state space for tracking of the robot in those regions. But such a transformation will also not promote formation of a dense cluster of samples

thereby impairing accurate pose estimation. The choice of the transformation function, thus, regulates the tradeoff between accuracy and robustness. We can, however, gain in accuracy without compromising much on robustness if we can devise a method for reliable estimation of the closeness of a particle to the true robot pose.

The method for weight computation that we propose uses a reliable way of assessing the closeness of a particle to the true, but unknown, pose. The transformation function that we use is acutely nonlinear making pose estimates possible in an accurate manner. The synergy resulting from the use of a reliable assessment method and the acutely nonlinear transformation endows an MCL using the proposed method of weight computation with the capability to continuously track the pose of a robot over fairly long paths in an accurate, robust and computationally efficient manner. It is to be specifically kept in mind that the proposed method has performed well on maps that were not handcrafted or produced by CAD software and that were not free of "noisy" features.

6.3 Future directions

The first step in our proposed method for map building consists in fitting line-segments to a set of closely-spaced range measurements by an optimization process that minimizes the departure of the line-segment from the measurements in a least-squares sense [94]. For ease of formulation, we ignored the noise associated with the range measurement. In reality, there is uncertainty associated with each of the noisy range sensor measurements, which introduces itself as uncertainty in the parameters of the fitted line-segments. Moreover, the improved estimate of each of the scan poses provided by the Lu-Milios method [71] is associated with an uncertainty. This uncertainty also contributes to the uncertainty in the parameters of the extracted line-segments when referred to the world coordinate system. The formulation of the proposed method could be augmented by taking these sources of uncertainties into account and propagating them into the estimates of the final map segments. Another interesting work could be the formulation of a method for reconstruction of the entire robot trajectory—a solution to the full SLAM problem like the Lu-Milios method [71]— but using line segments extracted from the laser range scans instead of the raw scans themselves like in [120].

The offline map building method that we propose uses laser range data collected by a robot. The robot is manually driven or is teleoperated through the entire environment envisaged to be mapped in an initial phase. To ensure completeness of the resulting map, it is imperative that the robot has "seen" the entire environment and collected range data from every nook-and-corner of the environment. It is desirable that the robot explores its environment for data collection in a complete and time-efficient

manner [193]. If the robot has not visited the entire environment, it will not have range data of the unvisited areas and thus these areas will remain unmodeled in the final map. If the robot visits the same area more than once, it will have redundant data. This will not provide any new information about the environment and will only add to the time required for data collection and subsequent processing. The displacements between two consecutive scan poses from which range data are collected have important ramifications in the map completeness versus computation-time trade-off. If the displacement is large, there may not be sufficient overlap between the scans taken from the corresponding poses. This will make scan registration challenging and may also lead to features of the environment not getting modeled in the map, thereby resulting in an incomplete map. If scans are taken from nearby poses, there will be significant overlap between the scans resulting in redundant data. This will take away a lot of CPU time for extracting the map from the laser scan data. Thus, it would be interesting and worthwhile to work towards a strategy that will enable a mobile robot to judiciously explore an environment and capture optimum laser range data for map building.

In this thesis, we have proposed ways for quantitative assessment of the goodness of maps produced and have also highlighted the shortcomings of the proposed methods. We have noted that map assessment is a tricky proposition as the basis on which such assessments can be made is often application-specific. It is important to investigate this aspect further to come up with more practical, elegant and methodical approaches for map quality assessment. It should be possible to precisely quantify the goodness of a map when ground-truth data is available [194]. It may also be useful to investigate the feasibility of assessing a map in absence of ground-truth data, possibly along the lines of [130].

Preliminary studies revealed that the proposed method of weight computation for Monte Carlo localization on line-segment based maps holds promise for global localization as well. The particles, initially dispersed all over the map, were found to converge in the vicinity of the robot within a few time steps and correctly track the robot pose thereafter. However, extensive simulation and experimental studies will have to be carried out before making any claim as to its usefulness for global localization. It would also be worth investigating the extent to which the global localization method would gain from the standpoint of efficiency, if it is augmented with a strategy for adaptively varying the particle size [195] by eliminating redundant particles quickly during the transition from global localization to pose tracking phase. The problem of premature convergence that arises often in MCL will also have to be looked into [196]. Moreover, a method for efficient generation of particles such that they are uniformly dispersed but confined inside the map in the initial state also has to be formulated.

In the proposed method of importance weight computation, we extracted line-

segments from laser range data (referred to as scan line segments) and assessed how closely they fitted with the environment map. Like in the method of map-building, here too, we did not consider the uncertainty associated with the range measurements that would otherwise have enabled us to model the uncertainty in the parameters of the extracted scan line segments. If uncertainties in the parameters of both scan line segments and map segments are considered, it may be possible to estimate the importance weights in a more reliable manner, provided of course that the weight computation method is formulated accordingly. This would lead to added robustness and accuracy of localization and would make the proposed map building and localization methods more useful and efficacious for deployment in real-life applications.

References

[1] K. Čapek, *R. U. R.* Dover Publications, 2001.

[2] I. Asimov, *I, Robot.* Doubleday, 1950.

[3] S. Bensalem, M. Gallien, F. Ingrand, I. Kahloul, and N. Thanh-Hung, "Designing autonomous robots," *Robotics Automation Magazine, IEEE*, vol. 16, no. 1, pp. 67–77, March 2009.

[4] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots.* Bradford Company, 2004.

[5] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, "Robocup: A challenge problem for AI." *AI Magazine*, vol. 18, no. 1, pp. 73–85, 1997.

[6] M. Bajracharya, M. Maimone, and D. Helmick, "Autonomy for Mars Rovers: Past, present, and future," *Computer*, vol. 41, no. 12, pp. 44–50, Dec 2008.

[7] G. Seetharaman, A. Lakhotia, and E. Blasch, "Unmanned vehicles come of age: The DARPA grand challenge," *Computer*, vol. 39, no. 12, pp. 26–29, Dec 2006.

[8] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA grand challenge: Research articles," *J. Robot. Syst.*, vol. 23, no. 9, Sep. 2006.

[9] C. Urmson, J. Anhalt, D. Bartz, M. Clark, T. Galatali, A. Gutierrez, S. Harbaugh, J. Johnston, H. Kato, P. L. Koon, W. Messner, N. Miller, A. Mosher, K. Peterson, C. Ragusa, D. Ray, B. K. Smith, J. M. Snider, S. Spiker, J. C. Struble, J. Ziglar, and W. R. L. Whittaker, "A robust approach to high-speed navigation for unrehearsed desert terrain," *Journal of Field Robotics*, vol. 23, no. 8, pp. 467–508, August 2006.

[10] C. Urmson, J. Anhalt, H. Bae, J. A. D. Bagnell, C. R. Baker, R. E. Bittner, T. Brown, M. N. Clark, M. Darms, D. Demitrish, J. M. Dolan, D. Duggins, D. Ferguson , T. Galatali, C. M. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, M. Likhachev , B. Litkouhi, A. Kelly , M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y.-W. Seo, S. Singh, J. M. Snider, J. C. Struble, A. T. Stentz, M. Taylor , W. R. L. Whittaker, Z. Wolkowicki, W. Zhang, and J. Ziglar, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, vol. 25, no. 8, pp. 425–466, June 2008.

[11] C. Urmson and W. R. L. Whittaker, "Self-driving cars and the urban challenge," *IEEE Intelligent Systems*, vol. 23, no. 2, pp. 66–68, April 2008.

[12] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma, "Emergency response to the nuclear accident at the Fukushima Daiichi nuclear power plants using mobile rescue robots," *Journal of Field Robotics*, vol. 30, no. 1, pp. 44–63, 2013.

[13] D. Guan, L. Yan, Y. Yang, and W. Xu, "A small climbing robot for the intelligent inspection of nuclear power plants," in *Information Science and Technology (ICIST), 2014 4th IEEE International Conference on*, April 2014, pp. 484–487.

[14] V. Michal, "14—remote operation and robotics technologies in nuclear decommissioning projects," in *Nuclear Decommissioning*, ser. Woodhead Publishing Series in Energy, M. Laraia, Ed. Woodhead Publishing, 2012, pp. 346–374.

[15] D. Seward and M. Bakari, "The use of robotics and automation in nuclear decommissioning." in *Proceedings of the 22nd international symposium on automation and robotics in construction*, 2005.

[16] L. Briones, P. Bustamante, and M. A. Serna, "Robicen: A wall-climbing pneumatic robot for inspection in nuclear power plants," *Robotics and Computer–Integrated Manufacturing*, vol. 11, no. 4, pp. 287–292, 1994.

[17] A. Shukla and H. Karki, "A review of robotics in onshore oil-gas industry," in *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, Aug 2013, pp. 1153–1160.

[18] X. Li, W. Yu, X. Lin, and S. Iyengar, "On optimizing autonomous pipeline inspection," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 223–233, Feb 2012.

[19] J. Hertzberg and F. Kirchner, "Landmark-based autonomous navigation in sewerage pipes," in *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on*, Oct 1996, pp. 68–73.

[20] P. K. Pal, K. Jayarajan, D. D. Ray, M. Singh, V. Mahadev, B. Sony, V. K. Shrivastava, A. N. Jha, R. V. Sakrikar, D. Mishra, V. V. Agashe, and P. Taktawala, "A mobile robot that removed and disposed ammunition boxes," *Current science*, vol. 92, no. 12, pp. 1673–1677, 2007.

[21] C. Pingyuan, Y. Fuzhan, and C. Hutao, "Research on autonomous navigation of lunar rovers for the moon exploration," in *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, Dec 2006, pp. 1042–1047.

[22] B. Muirhead, "Mars rovers, past and future," in *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, vol. 1, March 2004.

[23] R. Washington, K. Golden, J. Bresina, D. Smith, C. Anderson, and T. Smith, "Autonomous rovers for Mars exploration," in *Aerospace Conference, 1999. Proceedings. 1999 IEEE*, vol. 1, 1999, pp. 237–251 vol.1.

[24] J. Charles, "The lunar rover initiative: breaking ground, laying foundations," *IEEE Expert*, vol. 10, no. 6, pp. 5–6, Dec 1995.

[25] K. W. Weng and M. Abidin, "Design and control of a quad-rotor flying robot for aerial surveillance," in *Research and Development, 2006. SCOReD 2006. 4th Student Conference on*, June 2006, pp. 173–177.

[26] S. A. Johnson and J. M. Vallely, "A portable aerial surveillance robot," in *Proc. SPIE*, vol. 6201, 2006.

[27] C. Yuan, Y. Zhang, and Z. Liu, "A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques," *Canadian Journal of Forest Research*, vol. 45, no. 7, pp. 783–792, 2015.

[28] L. Merino, F. Caballero, J. Martnez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3–4, pp. 165–184, 2006.

[29] L. Merino, F. Caballero, J. R. Martínez-de Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1–4, pp. 533–548, 2012.

[30] M. Jacobi, "Autonomous inspection of underwater structures," *Robotics and Autonomous Systems*, vol. 67, pp. 80–86, 2015, advances in Autonomous Underwater Robotics.

[31] D. Miller, "Design of a small, cheap UUV for under-ship inspection and salvage," in *Autonomous Underwater Vehicle Technology, 1996. AUV '96., Proceedings of the 1996 Symposium on*, Jun 1996, pp. 18–20.

[32] K. Tanigaki, T. Fujiura, A. Akase, and J. Imagawa, "Cherry-harvesting robot," *Computers and Electronics in Agriculture*, vol. 63, no. 1, pp. 65–72, 2008, Special issue on Bio-robotics.

[33] T. Bak and H. Jakobsen, "Agricultural robotic platform with four wheel steering for weed detection," *Biosystems Engineering*, vol. 87, no. 2, pp. 125–136, 2004.

[34] E. van Henten, J. Hemming, B. van Tuijl, J. Kornet, J. Meuleman, J. Bontsema, and E. van Os, "An autonomous robot for harvesting cucumbers in greenhouses," *Autonomous Robots*, vol. 13, no. 3, pp. 241–258, 2002.

[35] N. Singh, P. Sarngadharan, and P. K. Pal, "AGV scheduling for automated material distribution: a case study," *Journal of Intelligent Manufacturing*, vol. 22, no. 2, pp. 219–228, 2011.

[36] B. R. Sarker and S. S. Gurav, "Route planning for automated guided vehicles in a manufacturing facility," *International Journal of Production Research*, vol. 43, no. 21, pp. 4659–4683, 2005.

[37] A. Langevin, D. Lauzon, and D. Riopel, "Dispatching, routing, and scheduling of two automated guided vehicles in a flexible manufacturing system," *International Journal of Flexible Manufacturing Systems*, vol. 8, no. 3, pp. 247–262, 1996.

[38] M. Takahashi, T. Suzuki, F. Cinquegrani, R. Sorbello, and E. Pagello, "A mobile robot for transport applications in hospital domain with safe human detection algorithm," in *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, Dec 2009, pp. 1543–1548.

[39] C. A. A. Calderon, E. R. Mohan, and B. S. Ng, "Development of a hospital mobile platform for logistics tasks," *Digital Communications and Networks*, vol. 1, no. 2, pp. 102–111, 2015.

[40] B. Coltin and M. Veloso, "Online pickup and delivery planning with transfers for mobile robots," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 5786–5791.

[41] E. Prassler, M. Strobel, M. Strobel, and T. Kämpke, "Mobile robots in office logistics," in *In Proceedings of the 27th International Symposium on Industrial Robots*, 1996, pp. 153–159.

[42] I. R. Nourbakhsh, R. Powers, and S. Birchfield, "DERVISH—an office-navigating robot." *AI Magazine*, vol. 16, no. 2, pp. 53–60, 1995.

[43] F. Vaussard, J. Fink, V. Bauwens, P. Rtornaz, D. Hamel, P. Dillenbourg, and F. Mondada, "Lessons learned from robotic vacuum cleaners entering the home ecosystem," *Robotics and Autonomous Systems*, vol. 62, no. 3, pp. 376–391, 2014.

[44] D. Fischinger, P. Einramhof, K. Papoutsakis, W. Wohlkinger, P. Mayer, P. Panek, S. Hofmann, T. Koertner, A. Weiss, A. Argyros, and M. Vincze, "Hobbit, a care robot supporting independent living at home: First prototype and lessons learned," *Robotics and Autonomous Systems*, pp. –, 2014.

[45] R. Bemelmans, G. J. Gelderblom, P. Jonker, and L. de Witte, "Socially assistive robots in elderly care: A systematic review into effects and effectiveness," *Journal of the American Medical Directors Association*, vol. 13, no. 2, pp. 114–120.e1, 2012.

[46] J. Broekens, M. Heerink, and H. Rosendal, "Assistive social robots in elderly care: a review," *Gerontechnology*, vol. 8, no. 2, 2009.

[47] T. Mei, M. Luo, X. Ye, J. Cheng, L. Wang, B. Kong, and R. Wang, "Chapter 2.4– Design and implementation of a service robot for elders," in *Household Service Robotics*, Y. X. Q. Wu, Ed. Oxford: Academic Press, 2015, pp. 83–93.

[48] J. L. Crowley, "Planning and execution of tasks for a domestic robot," *Robotics and Autonomous Systems*, vol. 5, no. 3, pp. 257–272, 1989.

[49] E. Falcone, R. Gockley, E. Porter, and I. Nourbakhsh, "The personal rover project: The comprehensive design of a domestic personal robot," *Robotics and Autonomous Systems*, vol. 42, no. 3–4, pp. 245–258, 2003, socially Interactive Robots.

[50] S. Alves, I. Silva, C. Ranieri, H. Filho, M. Caldeira, and R. Pegoraro, "A friendly mobile entertainment robot for disabled children," in *Biosignals and Biorobotics Conference (BRC), 2013 ISSNIP*, Feb 2013, pp. 1–6.

[51] W. Burgard, A. B. Cremers, D. Fox, D. Hhnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, no. 1–2, pp. 3–55, 1999.

[52] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hhnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *In Proceedings of IEEE International Conference on Robotics and Automation (ICRA99*, 1999.

[53] I. R. Nourbakhsh, J. Bobenage, S. Grange, R. Lutz, R. Meyer, and A. Soto, "An affective mobile robot educator with a full-time job," *Artificial Intelligence*, vol. 114, no. 1–2, pp. 95–124, 1999.

[54] H. R. Everett, *Sensors for Mobile Robots: Theory and Application.* A. K. Peters, Ltd., 1995.

[55] J.-C. Latombe, *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[56] R. R. Murphy, *Introduction to AI Robotics.* MIT Press, 2000.

[57] J. Borenstein and Y. Koren, "The vector field histogram—fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 278–288, Jun 1991.

[58] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, Mar 1997.

[59] P. K. Pal and A. Kar, "Sonar-based mobile robot navigation through supervised learning on a neural net," *Autonomous Robots*, vol. 3, no. 4, pp. 355–374, 1996.

[60] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, 1999, pp. 341–346 vol.1.

[61] R. C. Coulter, "Implementation of the Pure Pursuit path tracking algorithm," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, January 1992.

[62] D. Filliat and J.-A. Meyer, "Map-based navigation in mobile robots: I. a review of localization strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 243–282, 2003.

[63] J.-A. Meyer and D. Filliat, "Map-based navigation in mobile robots: II. a review of map-learning and path-planning strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 283–317, 2003.

[64] H. Moravec and A. E. Elfes, "High-resolution maps from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985, pp. 116–121.

[65] A. Elfes, "A sonar-based mapping and navigation system," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, Apr 1986, pp. 1151–1156.

[66] C. Stachniss and W. Burgard, "Mapping and exploration with mobile robots using coverage maps," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, Oct 2003, pp. 467–472 vol.1.

[67] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and Autonomous Systems*, vol. 8, no. 1-2, pp. 47–63, 1991, special Issue Toward Learning Robots.

[68] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[69] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *Robotics Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, June 2006.

[70] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *Robotics Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, Sept 2006.

[71] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333–349, 1997.

[72] B. Sarkar, P. K. Pal, and D. Sarkar, "Building maps of indoor environments by merging line segments extracted from registered laser range scans," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 603–615, 2014.

[73] B. Sarkar, S. Saha, and P. K. Pal, "A novel method for computation of importance weights in Monte Carlo localization on line segment-based maps," *Robotics and Autonomous Systems*, vol. 74, Part A, pp. 51–65, 2015.

[74] F. Amigoni and M. Vailati, "A method for reducing redundant line segments in maps," in *Proceedings of the 4th European Conference on Mobile Robots, ECMR'09, September 23-25, 2009, Mlini/Dubrovnik, Croatia*, 2009, pp. 61–66.

[75] R. Lakaemper, "Simultaneous multi-line-segment merging for robot mapping using mean shift clustering," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct 2009, pp. 1654–1660.

[76] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1322–1328 vol.2.

[77] J.-S. Gutmann and D. Fox, "An experimental comparison of localization methods continued," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, 2002, pp. 454–459 vol.1.

[78] P. Espinace, A. Soto, and M. Torres-Torriti, "Real-time robot localization in indoor environments using structural information," in *Robotic Symposium, 2008. LARS '08. IEEE Latin American*, Oct 2008, pp. 79–84.

[79] T. Yaqub and J. Katupitiya, "Laser scan matching for measurement update in a particle filter," in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, Sept 2007, pp. 1–6.

[80] A. Gasparri, S. Panzieri, F. Pascucci, and G. Ulivi, "Genetic approach for a localisation problem based upon particle filters," in *Proc. of 8th Int. IFAC Symp. On Robot Control (SYROCO 2006)*, Bologna, Italy, 2006.

[81] R. Ravishankar, T. Bhaumik, T. Bandyopadhyay, M. Purkait, S. Jena, S. Mishra, S. Sharma, V. Agashe, K. Datta, B. Sarkar, C. Datta, D. Sarkar, and P. Pal, "Radiation mapping inside the bunkers of medium energy accelerators using a robotic carrier," *Applied Radiation and Isotopes*, vol. 80, pp. 103–108, 2013.

[82] R. O. Duda and P. E. Hart, *Pattern Classification and scene analysis.* John Wiley & Sons, Inc., 1973.

[83] T. Pavlidis and S. Horowitz, "Segmentation of plane curves," *Computers, IEEE Transactions on*, vol. C-23, no. 8, pp. 860–870, Aug 1974.

[84] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics," *Autonomous Robots*, vol. 23, no. 2, pp. 97–111, Aug. 2007.

[85] G. Borges and M.-J. Aldon, "Line extraction in 2D range images for mobile robotics," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 3, pp. 267–297, 2004.

[86] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson, "An optimized segmentation method for a 2D laser scanner applied to mobile robot navigation," in *Proceedings of the 3rd IFAC symposium on intelligent components and instruments for control applications*, 1997.

[87] S. Pfister, S. Roumeliotis, and J. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *Robotics and Automation, 2003 Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, Sept 2003, pp. 1304–1311 vol.1.

[88] K. O. Arras and R. Y. Siegwart, "Feature extraction and scene interpretation for map-based navigation and map building," in *Proc. of SPIE, Mobile Robotics XII*, 1997, pp. 42–53.

[89] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, Jun. 1981.

[90] D. Sack and W. Burgard, "A comparison of methods for line extraction from range data," in *In Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[91] A. Harati and R. Siegwart, "A new approach to segmentation of 2D range scans into linear regions," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, Oct 2007, pp. 2083–2088.

[92] C. Fernández, V. Moreno, B. Curto, and J. A. Vicente, "Clustering and line detection in laser range measurements," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 720–726, 2010.

[93] Y. Zhao and X. Chen, "Prediction-based geometric feature extraction for 2d laser scanner," *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 402–409, 2011.

[94] N. R. Draper and H. Smith, *Applied Regression Analysis*. John Wiley & Sons, Inc., 2005.

[95] B. K. Ray and K. S. Ray, "Determination of optimal polygon from digital curve using L_1 norm," *Pattern Recognition*, vol. 26, no. 4, pp. 505–509, 1993.

[96] ——, "A non-parametric sequential method for polygonal approximation of digital curves," *Pattern Recognition Letters*, vol. 15, no. 2, pp. 161–167, 1994.

[97] K. Wall and P.-E. Danielsson, "A fast sequential method for polygonal approximation of digitized curves," *Computer Vision, Graphics, and Image Processing*, vol. 28, no. 2, pp. 220 – 227, 1984.

[98] I. M. Anderson and J. Bezdek, "Curvature and tangential deflection of discrete arcs: A theory based on the commutator of scatter matrix pairs and its application to vertex detection in planar shape data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, no. 1, pp. 27–40, Jan 1984.

[99] H. Freeman and L. Davis, "A corner-finding algorithm for chain-coded curves," *Computers, IEEE Transactions on*, vol. C-26, no. 3, pp. 297–303, March 1977.

[100] D. Sarkar, "A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves," *Pattern Recogn. Lett.*, vol. 14, no. 12, pp. 959–964, Dec. 1993.

[101] C. H. Teh and R. T. Chin, "On the detection of dominant points on digital curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 8, pp. 859–872, Aug. 1989.

[102] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[103] P.-Y. Yin, "A new method for polygonal approximation using genetic algorithms," *Pattern Recognition Letters*, vol. 19, no. 11, pp. 1017–1026, 1998.

[104] ——, "Genetic algorithms for polygonal approximation of digital curves," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 13, no. 07, pp. 1061–1082, 1999.

[105] B. Sarkar, L. K. Singh, and D. Sarkar, "A genetic algorithm-based approach for detection of significant vertices for polygonal approximation of digital curves," *International Journal of Image and Graphics*, vol. 04, no. 02, pp. 223–239, 2004.

[106] P.-Y. Yin, "A tabu search approach to polygonal approximation of digital curves," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 02, pp. 243–255, 2000.

[107] ——, "Ant colony search algorithms for optimal polygonal approximation of plane curves," *Pattern Recognition*, vol. 36, no. 8, pp. 1783–1797, 2003.

[108] ——, "A discrete particle swarm algorithm for optimal polygonal approximation of digital curves," *Journal of Visual Communication and Image Representation*, vol. 15, no. 2, pp. 241–260, 2004.

[109] B. Sarkar, "An efficient method for near-optimal polygonal approximation based on differential evolution," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 22, no. 06, pp. 1267–1281, 2008.

[110] S.-C. Huang, "Polygonal approximation using an artificial bee colony algorithm," *Mathematical Problems in Engineering*, 2015.

[111] C. Ichoku, B. Deffontaines, and J. Chorowicz, "Segmentation of digital plane curves: A dynamic focusing approach," *Pattern Recognition Letters*, vol. 17, no. 7, pp. 741–750, 1996.

[112] J.-H. Horng and J. T. Li, "A dynamic programming approach for fitting digital planar curves with line segments and circular arcs," *Pattern Recognition Letters*, vol. 22, no. 2, pp. 183–197, 2001.

[113] B. Sarkar, L. K. Singh, and D. Sarkar, "Approximation of digital curves with line segments and circular arcs using genetic algorithms," *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2585–2595, 2003.

[114] B. Sarkar, S. Roy, and D. Sarkar, "Hierarchical representation of digitized curves through dominant point detection," *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2869–2882, 2003.

[115] S. Thrun, "Exploring artificial intelligence in the new millennium," G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann Publishers Inc., 2003, ch. Robotic Mapping: A Survey, pp. 1–35.

[116] M. A. Movafaghpour and E. Masehian, "Poly line map extraction in sensor-based mobile robot navigation using a consecutive clustering algorithm," *Robotics and Autonomous Systems*, vol. 60, no. 8, pp. 1078–1092, 2012.

[117] J. Gonzalez, A. Ollero, and A. Reina, "Map building for a mobile robot equipped with a 2D laser rangefinder," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, May 1994, pp. 1904–1909 vol.3.

[118] L. Zhang and B. Ghosh, "Line segment based map building and localization using 2D laser rangefinder," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 3, 2000, pp. 2538–2543 vol.3.

[119] F. Amigoni, G. Fontana, and F. Garigiola, "A method for building small-size segment-based maps," in *Distributed Autonomous Robotic Systems 7*, M. Gini and R. Voyles, Eds. Springer Japan, 2006, pp. 11–20.

[120] J. Elseberg, R. T. Creed, and R. Lakaemper, "A line segment based system for 2D global mapping," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3924–3931.

[121] F. Amigoni, S. Gasparini, and M. Gini, "Building segment-based maps without pose information," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1340–1359, July 2006.

[122] F. Amigoni and S. Gasparini, "Analysis of methods for reducing line segments in maps: Towards a general approach," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept 2008, pp. 2896–2901.

[123] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *Information Theory, IEEE Transactions on*, vol. 21, no. 1, pp. 32–40, Jan 1975.

[124] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, May 2002.

[125] Y. Cheng, "Mean shift, mode seeking, and clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 17, no. 8, pp. 790–799, Aug 1995.

[126] K.-L. Wu and M.-S. Yang, "Mean shift-based clustering," *Pattern Recognition*, vol. 40, no. 11, pp. 3035–3052, 2007.

[127] S.-J. Chang-Chien, W.-L. Hung, and M.-S. Yang, "On mean shift-based clustering for circular data," *Soft Comput.*, vol. 16, no. 6, pp. 1043–1060, Jun. 2012.

[128] I. J. Cox, "Blanche—an experiment in guidance and navigation of an autonomous robot vehicle," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 2, pp. 193–204, Apr 1991.

[129] Y. L. Ip, A. B. Rad, K. M. Chow, and Y. K. Wong, "Segment-based map building using enhanced adaptive fuzzy clustering algorithm for mobile robot applications," *J. Intell. Robotics Syst.*, vol. 35, no. 3, pp. 221–245, Nov. 2002.

[130] R. Lakaemper, "A confidence measure for segment based maps," in *Proceedings of the 9th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS '09. ACM, 2009, pp. 211–216.

[131] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, Nov. 2009.

[132] F. Amigoni, S. Gasparini, and M. Gini, "Good experimental methodologies for robotic mapping: A proposal," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 4176–4181.

[133] J. S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Computational Intelligence in Robotics and Automation, 1999. CIRA '99. Proceedings. 1999 IEEE International Symposium on*, 1999, pp. 318–325.

[134] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287–300.

[135] K. Konolige, "Large-scale map-making," in *Proceedings of the 19th National Conference on Artifical Intelligence*, ser. AAAI'04. AAAI Press, 2004, pp. 457–463.

[136] U. Frese, *Spatial Cognition IV. Reasoning, Action, Interaction: International Conference Spatial Cognition 2004, Frauenchiemsee, Germany, October 11–13, 2004, Revised Selected Papers*. Springer Berlin Heidelberg, 2005, ch. Treemap: An O(log n) Algorithm for Simultaneous Localization and Mapping, pp. 455–477.

[137] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Rob. Res.*, vol. 25, no. 12, pp. 1181–1203, 2006.

[138] E. Olson, J. J. Leonard, and S. J. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA*, 2006, pp. 2262–2269.

[139] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *In Proc. of Robotics: Science and Systems (RSS)*, 2007.

[140] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, 2010.

[141] A. Howard and N. Roy, "The Robotics Data Set Repository (Radish)," 2003. [Online]. Available: http://radish.sourceforge.net/

[142] C. Guerra and V. Pascucci, "Line-based object recognition using Hausdorff distance: From range images to molecular secondary structures," *Image Vision Comput.*, vol. 23, no. 4, pp. 405–415, Apr. 2005.

[143] F. Donoso-Aguirre, J.-P. Bustos-Slas, M. Torres-Torriti, and A. Guesalaga, "Mobile robot localization using the Hausdorff distance," *Robotica*, vol. 26, no. 2, pp. 129–141, Mar. 2008.

[144] F. Amigoni, M. Reggiani, and V. Schiaffonati, "An insightful comparison between experiments in mobile robotics and in science," *Autonomous Robots*, vol. 27, no. 4, pp. 313–325, 2009.

[145] A. Gasparri, S. Panzieri, F. Pascucci, and G. Ulivi, "Monte Carlo filter in mobile robotics localization: A clustered evolutionary point of view," *Journal of Intelligent and Robotic Systems*, vol. 47, no. 2, pp. 155–174, 2006.

[146] T. He and S. Hirose, "A global localization approach based on line-segment relation matching technique," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 95–112, 2012.

[147] J. Gonzalez, A. Stentz, and A. Ollero, "A mobile robot iconic position estimator using a radial laser scanner," *Journal of Intelligent and Robotic Systems*, vol. 13, no. 2, pp. 161–179, 1995.

[148] H. Sohn and B. Kim, "An efficient localization algorithm based on vector matching for mobile robots using laser range finders," *Journal of Intelligent and Robotic Systems*, vol. 51, no. 4, pp. 461–488, 2008.

[149] G. Shaffer, A. Stentz, W. Whittaker, and K. Fitzpatrick, "Position estimator for underground mine equipment," *Industry Applications, IEEE Transactions on*, vol. 28, no. 5, pp. 1131–1140, Sep 1992.

[150] A. Reina and J. Gonzalez, "A two-stage mobile robot localization method by overlapping segment-based maps," *Robotics and Autonomous Systems*, vol. 31, no. 4, pp. 213–225, 2000.

[151] H. J. Sohn and B. K. Kim, "A robust localization algorithm for mobile robots with laser range finders," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, April 2005, pp. 3545–3550.

[152] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," *J. Intell. Robotics Syst.*, vol. 18, no. 3, pp. 249–275, Mar. 1997.

[153] G. Weiss, C. Wetzler, and E. von Puttkamer, "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans," in *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 1, Sep 1994, pp. 595–601 vol.1.

[154] J.-S. Gutmann and C. Schlegel, "AMOS: comparison of scan matching approaches for self-localization in indoor environments," in *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on*, Oct 1996, pp. 61–67.

[155] L. Iocchi and D. Nardi, "Hough localization for mobile robots in polygonal environments," *Robotics and Autonomous Systems*, vol. 40, no. 1, pp. 43–58, 2002.

[156] B. Schiele and J. Crowley, "A comparison of position estimation techniques using occupancy grids," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, May 1994, pp. 1628–1634 vol.2.

[157] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, pp. 93–109, 2000.

[158] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[159] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 376–382, Jun 1991.

[160] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," in *In Proceedings of the Thirteenth National Conference on Artificial Intelligence, Menlo Park*. AAAI, AAAI Press/MIT Press, 1996, pp. 896–901.

[161] W. Burgard, D. Fox, and D. Hennig, "Fast grid-based position tracking for mobile robots," in *In Proc. of the 21th German Conference on Artificial Intelligence*. Springer Verlag, 1997, pp. 289–300.

[162] W. Burgard, A. Derr, D. Fox, and A. Cremers, "Integrating global position estimation and position tracking for mobile robots: the dynamic Markov localization approach," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 2, Oct 1998, pp. 730–735.

[163] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige, "An experimental comparison of localization methods," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 2, Oct 1998, pp. 736–743 vol.2.

[164] J.-S. Gutmann, "Markov-Kalman localization for mobile robots," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2, 2002, pp. 601–604 vol.2.

[165] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, Feb 2002.

[166] D. Fox, S. Thrun, F. Dellaert, and W. Burgard, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2000, ch. Particle filters for mobile robot localization.

[167] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1–2, pp. 99–141, 2001.

[168] T.-B. Kwon, J.-H. Yang, J.-B. Song, and W. Chung, "Efficiency improvement in Monte Carlo localization through topological information," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 424–429.

[169] J.-S. Lee and W. Chung, "Robust mobile robot localization in highly non-static environments," *Autonomous Robots*, vol. 29, no. 1, pp. 1–16, 2010.

[170] H. Köse and H. L. Akın, "The Reverse Monte Carlo localization algorithm," *Robot. Auton. Syst.*, vol. 55, no. 6, pp. 480–489, Jun. 2007.

[171] J. Rowekamper, C. Sprunk, G. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 3158–3164.

[172] L. Moreno, J. M. Armingol, S. Garrido, A. De La Escalera, and M. A. Salichs, "A genetic algorithm for mobile robot localization using ultrasonic sensors," *Journal of Intelligent and Robotic Systems*, vol. 34, no. 2, pp. 135–154, Jun. 2002.

[173] L. Ronghua and H. Bingrong, "Coevolution based adaptive Monte Carlo localization (CEAMCL)," *International Journal of Advanced Robotic Systems*, vol. 1, no. 3, pp. 183–190, 2004.

[174] M. Lisowski, Z. Fan, and O. Ravn, "Differential evolution to enhance localization of mobile robots," in *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, June 2011, pp. 241–247.

[175] L. Moreno, S. Garrido, and M. Munoz, "Evolutionary filter for robust mobile robot global localization," *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 590–600, 2006.

[176] F. Martín, L. Moreno, D. Blanco, and M. L. Munoz, "Kullback-Leibler divergence-based global localization for mobile robots," *Robotics and Autonomous Systems*, vol. 62, no. 2, pp. 120–130, 2014.

[177] M. Bashiri, H. Vatankhah, and S. Shiry Ghidary, "Hybrid adaptive differential evolution for mobile robot localization," *Intelligent Service Robotics*, vol. 5, no. 2, pp. 99–107, 2012.

[178] A. Vahdat, N. NourAshrafoddin, and S. Ghidary, "Mobile robot global localization using differential evolution and particle swarm optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, Sept 2007, pp. 1527–1534.

[179] A. Gasparri, S. Panzieri, F. Pascucci, and G. Ulivi, "A spatially structured genetic algorithm over complex networks for mobile robot localisation," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 4277–4282.

[180] A. Gasparri and M. Prosperi, "A bacterial colony growth algorithm for mobile robot localization," *Autonomous Robots*, vol. 24, no. 4, pp. 349–364, 2008.

[181] M. Mirkhani, R. Forsati, A. M. Shahri, and A. Moayedikia, "A novel efficient algorithm for mobile robot localization," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 920–931, 2013.

[182] J. Moravec and P. Pošík, "A comparative study: the effect of the perturbation vector type in the differential evolution algorithm on the accuracy of robot pose and heading estimation," *Evolutionary Intelligence*, vol. 6, no. 3, pp. 171–191, 2014.

[183] M.-P. Dubuisson and A. Jain, "A modified Hausdorff distance for object matching," in *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1, Oct 1994, pp. 566–568 vol.1.

[184] S. Saha, B. Sarkar, and P. K. Pal, "Monte Carlo-based pose tracking on maps represented with line segments," in *Proceedings of the 2015 International Conference on Advances In Robotics*, ser. AIR '15. New York, NY, USA: ACM, 2015, pp. 62:1–62:6.

[185] O. Wulf, B. Wagner, and M. Khalaf-allah, "Using 3D data for Monte Carlo localization in complex indoor environments," in *Proceedings of the 2nd Bi-Annual European Conference on Mobile Robots, ECMR*, 2005, pp. 170–175.

[186] A. F. M. Smith and A. E. Gelfand, "Bayesian statistics without tears: A sampling–resampling perspective," *The American Statistician*, vol. 46, no. 2, pp. 84–88, 1992.

[187] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107–113, Apr 1993.

[188] D. B. Rubin, *Bayesian Statistics-3*. Oxford University Press, 1988, ch. Using the SIR Algorithm to Simulate Posterior Distributions.

[189] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, 1987, pp. 14–21.

[190] J. Beveridge and E. Riseman, "How easy is matching 2D line models using local search?" *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 6, pp. 564–579, Jun 1997.

[191] M. Mazuran and F. Amigoni, "Matching line segment scans with mutual compatibility constraints," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 4298–4303.

[192] F. Amigoni, M. Reggiani, and V. Schiaffonati, "Simulations used as experiments in autonomous mobile robotics," in *IEEE International Conference on Robotics and Automation, ICRA, Workshop on "The Role of Experiments in Robotics Research"*, 2010.

[193] H. H. González-Baños and J.-C. Latombe, "Navigation strategies for exploring indoor environments." *International Journal of Robotic Research*, vol. 21, no. 10–11, pp. 829–848, 2002.

[194] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. Sorrenti, and P. Taddei, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Autonomous Robots*, vol. 27, no. 4, pp. 353–371, 2009.

[195] D. Fox, "KLD-sampling: Adaptive particle filters," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[196] G. Kootstra and B. de Boer, "Tackling the premature convergence problem in Monte-Carlo localization," *Robotics and Autonomous Systems*, vol. 57, no. 11, pp. 1107 – 1118, 2009.