A TOPOLOGICAL APPROACH TO ONLINE AUTONOMOUS MAP BUILDING FOR MOBILE ROBOT NAVIGATION

SYEDA NUSRAT FERDAUS

A Topological Approach to Online Autonomous Map Building for Mobile Robot Navigation

by

© Syeda Nusrat Ferdaus

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

October 2008

St. John's Newfoundland

Abstract

This thesis presents an online autonomous mobile robot exploration and navigation strategy. An appropriate environmental representation is an essential part of an efficient navigation system. We choose a topological map representation, where the world is represented by a set of omnidirectional images captured at each node with edges joining the nodes. Topological maps are memory-efficient and enable fast and simple path planning towards a specified goal. Using a laser range finder and an omnidirectional camera, an online topological representation of the environment is developed; although the navigation process relies only on the omnidirectional camera. We choose to use an omnidirectional camera, because it gives a 360 horizontal field-of-view and offers other advantages, such as increased robustness to occlusion, rich information content, etc. A view classifier based on global image comparison technique is used in order to avoid the possibility of creating a node in the same or nearby location where another node was created in the topological map.

A robot navigation system is presented which is based on visual information only. The visual homing mechanism is used to move the robot from one node to another in the topological map. Visual homing can be defined as the ability to return to a goal location by performing some kind of matching between the image taken while at the goal and the current view image. Path planning algorithm is implemented for successful vision-based navigation. All the experiments are done in an office environment using a Pioneer 3AT mobile robot. The topological map is built in real time on board the robot, thus making the system autonomous.

Acknowledgements

I would like to express my gratitude to all those who helped me to complete this thesis. I am greatly indebted to my supervisors Dr. George Mann, Dr. Andrew Vardy and Dr. Raymond Gosine for their guidance and continuing support, without which I could not have completed this programme. They have guided me with great care throughout my graduate studies, offered me words of wisdom when needed and encouraged me to explore new areas. I am also thankful to all my colleagues at the Intelligent Systems Lab (ISLAB), their friendship means a lot to me. I would like to thank Dr. Andrew Vardy and Dave Churchill for providing their code.

Finally, heartfelt thanks are due to my parents and my husband; their unconditional love and support made it possible for me to complete this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

In recent times, mobile robots are being used to perform various tasks in many sectors, including the automobile industry, health care, domestic households, office environments, underwater experiments, research laboratories, etc [5, 20, 33]. They are also used as tour guides in museums [6]. Mobile robots are a good choice for those tasks which are considered dangerous for humans, such as - working in coal mines, nuclear reactors, volcanos, etc [3, 59]. Navigation is important for these mobile robots, because many tasks involve going from one location to another or moving some material to a certain place in the environment. An appropriate map representation of the environment is a prerequisite for successful navigation of mobile agents. The most common method of representing the environment is by building a geometric map. In geometric maps, the environment is mapped by recording the distances and angles of all perceivable objects in the global reference frame. On the other hand, topological

maps represent the environment with a graph, with nodes representing significant places in the environment and edges representing traversable paths between nodes. The exact geometric location of a node is not required for this type of mapping.

Our main focus is to build a map that facilitates proper navigation; precise geometric estimate of position is not required for our approach. We choose to use a topological map, where the world is represented by a set of omnidirectional images captured at each node with edges joining the nodes. In our approach, the map is built autonomously using both laser and visual information. Laser data is used to find promising directions for exploration. On the other hand, vision is used to distinguish between different places in the environment.

Mobile robot navigation is a vast area of research; a number of different approaches have been taken to solve the problem. We choose to use visual navigation in our system. Visual navigation schemes can be seen in different species in nature. Bee and ants are able to go to their food sources and then return to their home despite having a very small brain; they can navigate long distances using only visual information [65]. Visual navigation therefore provides a number of biological examples; different ideas and techniques for robot navigation can be drawn from them.

We use an omnidirectional camera for our system. The main reason behind the use of omnidirectional vision is that of obtaining a wide field-of-view; this makes the system robust to small changes in the environment [66]. An omnidirectional system provides rotational invariance in the field of view, i.e. the same area of the environment is captured independent of the orientation of the camera [62]. Another great advantage is that an image, captured at a certain location with an omnidirectional camera, contains enough information to distinguish it from another omnidirectional

image captured from a nearby location. On the other hand, images captured with a conventional camera can not provide a 360^0 view, so the matching between a certain image and all the other database images becomes difficult, sometimes it may result into erroneous matching.

1.2 Research Objectives

Exploration and navigation are two very active areas of research in mobile robotics. Our main objective in this research work is to build a topological map using a mobile robot without any intervention by a human operator i.e. an autonomous exploration system. In other words, we want to determine whether an autonomous robot can actually form (i.e. learn) a topological map of its environment - especially when the environment is ambiguous and contains loops. Specifically, we want to build a topological map that satisfies the following requirements: (a) simple and easy to build, (b) can be built online in realtime, (c) does not utilize a large amount of memory and (d) uses only laser data and visual information. We also want to determine if such a map can be used for vision-based navigation, i.e. movement to arbitrary locations within the topological map.

1.3 Contribution

In our system, the robot performs collision free exploration and topological map construction, path planning and navigation. Our approach does not depend on any artificial markers or any modifications to the environment, instead our system is

able to perform in unmodified office environments. The specific contributions of this research work are as follows:

1. We have been able to build an autonomous exploration system with a topological environmental representation, using a mobile robot without any manual control. We have successfully implemented visual homing process in order to build an efficient topological map where it is guaranteed that another node will not be created on the same or nearby location where a previous node was created. A solution to the loop closing problem is also provided.

2. We have developed an algorithm for finding promising exploration directions in the environment.

3. We have shown successful path planning and vision-based navigation with our topological map.

Some preliminary results of this research work have appeared in [19].

1.4 Organization

Chapter 2 provides the background knowledge necessary for the research work performed in this thesis. We review different environmental representations, various image matching techniques, both global and local, in this chapter. Discussions on visual homing, path planning and topological navigation are also provided.

Chapter 3 details some image matching techniques used in the image-based localization experiments. A discrete Bayes filter is used in the localization scheme, we

also talk about the motion model and the measurement model used in the experiments. The image-based localization experiments are done in order to compare the performance of image matching techniques in different situations.

Chapter 4 details our system architecture for topological map building and navigation. Different parts of the system, such as - exploration strategies, homing mechanism, rotation estimation, path planning, vision-based navigation, obstacle avoidance, etc. are described in this chapter.

Chapter 5 provides the experimental results obtained using a real robot. Limitations of our method are also discussed here.

Chapter 6 presents our conclusions and directions for possible future research.

Chapter 2

Background

Mobile robotics is a vast area of research; there are many directions to explore, such as robot localization, navigation, map building, cognitive robotics, egomotion, simultaneous localization and mapping (SLAM), etc. However, in this research work, we focus on developing a topological representation of the environment and performing efficient navigation using the topological map.

In this chapter, we review basic components of a mobile robot exploration and navigation system. Different types of map representations are discussed in section 2.1. Section 2.2 is about various visual homing mechanisms; we review some path planning methods in section 2.3. Different topological navigation algorithms are discussed in section 2.4.

2.1 Environmental Representation

The ability of a mobile agent to navigate depends to a great extent upon its appropriate environmental representation. Environmental representation can be defined as

the map or model created by a mobile agent to interpret the environment around it. Generally, an environment is represented in one of the three ways given below,

1. Geometric Maps
2. Topological Maps
3. Hybrid Maps

There have been a lot of research works in mobile robotics and artificial intelligence systems using the above representations to map an environment. A review of some research works is provided below, which includes works on geometric maps and topological maps; research works that use hybrid maps to represent the environment are also reviewed.

2.1.1 Geometric Maps

Geometric maps represent the environment with metric information and landmarks; the environment is mapped based on the objects and the distances among them. The representation is based on a global frame of reference, the positions and orientations of all objects (including the robot) are given in this global reference frame. A variety of sensors are used to calculate the distances and angles from the robot to the objects in the world; thus this type of mapping depends on the accuracy of sensory observations i.e. erroneous sensory data may result in an erroneous geometric map [58]. Most of the earlier research work on robotics used geometric maps to represent the environment. Occupancy grid maps can be referred to as a classical representation of geometric maps [58]. In this type of mapping, the environment is modeled as a grid in which each

position or cell in the grid is assigned as either free space or occupied. Probabilistic models can also be used instead of binary values to represent uncertainties. Moravec [43] built a probabilistic, occupancy grid map from integrated sonar and vision data. Each cell in the grid contained the probability value of the space being occupied. The probability value of each cell was based on the accumulated sensor readings. Elfes [18] used a similar approach to generate a probabilistic grid map in unknown and unstructured surroundings using sonar sensors. Elfes implemented a robust method to interpret the sensory observations in such a way that the uncertainties and errors in the data could be reduced properly.

Thrun *et al.* [57] used geometric maps to solve the problem of concurrent map building and localization (this problem is also known as Simultaneous Localization and Mapping or *SLAM*). They implemented a practical maximum likelihood algorithm for generating the most probable map from the data; i.e. the algorithm alternates between localization and mapping and thus refines both the robot's location and the object locations in the map. Location of the robot is estimated in the E-step based on the currently available map and the M-step estimates a maximum likelihood map based on the locations computed in the E-step. In a later work, Thrun [55] presented a probabilistic approach for building geometric maps online; in this approach a team of robots was used to build the map. The maps were generated using a fast maximum likelihood approach under the most recent sensor measurement. To reduce the error in pose estimation, Thrun proposed a second estimator, i.e. Monte Carlo localization was implemented. The author claimed that by combining these two methods, the resulting algorithm could cope with large odometry errors typically found when mapping an environment with cycles. A method for developing three-dimensional maps

was also presented. Davison and Murray [15] developed a vision-based SLAM process based on the extended Kalman Filter (EKF). Visual landmarks were detected using the Harris corner detector [27] with a high-performance stereo head. An autonomous real-time implementation in a complex environment was presented. Castellanos *et al.* [9] presented a different SLAM implementation which was based on a multisensor system composed of a 2D laser range finder and a camera. Landmarks were detected using the laser range finder and then vision was used to obtain additional information about those landmarks. Se *et al.* [51] also developed a vision-based SLAM process which used scale-invariant image features as landmarks. Using a trinocular stereo system, 3D landmarks were obtained; then they are used for concurrent robot pose estimation and 3D map building. SLAM is a huge research area, a significant amount of work has been done in the past and research works are still being done in this field; as a result, it is not possible to present an exhaustive review on SLAM in the limited frame work of this thesis. Further knowledge on SLAM can be gathered from the book by Thrun *et al.* [58].

2.1.2 Topological Maps

A different representation from the geometric one is the topological map, which is less concerned with the exact geometry of the environment. Topological representation can be thought of as a coarse graph-like representation, where nodes correspond to significant places in the environment; these nodes are interconnected by traversable paths or edges [64, 21, 34]. An edge between two nodes is created only if the path is traversable i.e. if the robot can move from one node to the other one. Kuipers [34]

was one of the pioneer researchers in using the topological representations in the field of artificial intelligence. He developed a model called *TOUR* model which was based on the cognitive mapping humans use. Using the *TOUR* model, the author showed how spatial knowledge is stored in cognitive maps, how new information is integrated, how routes from one place to another can be found and how all the information can be used to solve various other problems. Later, Kuipers and Byun [36, 37] developed a deterministic topological model which contained distinctive places and edges connecting them; they used a hill-climbing search algorithm to select distinctive places or landmarks. In their model, each distinctive place contained local metrical information in order to facilitate the navigation process. Their assumption was that the current state could be determined from local information or the distance traversed from the previous state; as a result, the performance of their method was not satisfactory in cases where multiple actions were required to determine robot's current state. Instead of using a real mobile agent, they performed the experiments with a simulated robot in a variety of 2-D environments, their model was able to build an accurate map of an unknown environment even in the presence of sensory error.

Koenig and Simmons [30, 31] came up with a different approach for building topological maps. In their method, the robot was provided with some topological constraints that are easily obtainable by humans; then the robot would learn other necessary information while performing navigation or other tasks. Shatkay and Kaelbling [52] extended the approach of Koenig and Simmons [30]. However, they did not provide the robot with any local topological information beforehand, instead, they used local odometry information to build the map by applying an extended version of the Baum-Welch algorithm; local landmark observations were used to disambiguate

different locations. They used hidden Markov models (*HMMs*) for robot navigation.

Franz *et al.* [21] developed a topological representation of the environment using only a visual sensor. Their work extends the view graph approach of Schölkopf and Mallot [50] where they presented a vision-based scheme for building view graphs containing information on the topological and directional structure of the world. Franz *et al.* [21] implemented a simple view classifier which was used to select snapshot images or significant places to create nodes of the topological map. The view classifier was also used to check if the current view image was similar to any known node image of the topological map; therefore preventing the creation of another node in the same or nearby location of a node. In the case of the robot arriving at a nearby location of an already created node, a scene-based homing strategy was used to move the robot to that known node. Similarly, Goedeme *et al.* [24] generated a topological map using only an omnidirectional camera as the visual sensor. The map was built from a sequence of training images captured by manually driving the mobile agent around the environment. They implemented an image comparison technique which is a combination of two wide baseline features, namely a rotation reduced and color enhanced form of *SIFT* features [38] and the invariant column segment features [25]. They also implemented loop closing which was based on Dempster-Shafer probability theory. Winters [66] developed a topological map for mobile robot navigation using omnidirectional vision. However, the nodes within the environment were specified by a human operator. As a result, the robot was not completely autonomous.

Choset and Nagatani [11] implemented a topological SLAM process which was able to localize the robot with a partially explored map. The assumption of the mapping strategy was that the obstacles in the environment were planar extrusions

into three-dimensions. The mapping strategy used in their work was the generalized Voronoi graph (GVG), which was a map embedded in the robots free space. The intent of the GVG is to capture the topologically salient features of the free space. They presented some low-level control laws to generate the GVG edges and nodes using line-of-sight range data. Rybski *et al.* [48] used an appearance-based approach to SLAM implementation for very small, resource-limited robots having poor sensory information; the robot used in their work was equipped with a single monocular camera. Images were captured using a monocular camera and the Lucas-Kanade-Tomasi (KLT) feature tracker was used to find the best match between two images. The iterated form of the Extended Kalman Filter (IEKF) was used to estimate the coordinates of image locations; they tested their algorithm with simulated and real world data. Porta and Krose [45] developed a system that was able to perform SLAM using a multi-hypotheses tracker. The map was represented by a set of Gaussian mixtures with associated image features. The assumption in the mapping strategy was that the robot would move repetitively in the same environment. The map was constructed online and it was continuously refined as the robot moved through the environment which improved the localization of the mobile robot. Gross *et al.* [26] presented an omnivision-based SLAM approach which was able to localize a mobile robot in complex and dynamic environments. In their method, the current observation was described with a distributed coding, which used a set of the most similar reference observations. A generalized scheme for fusion of localization hypotheses from several state estimators with different levels of certainty was used. Several real-world localization experiments were performed to test their method.

2.1.3 Hybrid Maps

Both geometric and topological maps have some advantages, for this reason a number of researchers implemented an integrated hybrid map using aspect of both strategies. Among the earliest research works using the integrated map is an approach by Chatila and Laumond [10]. In their method, objects were denoted by polyhedra in a global coordinate frame; the free-space was decomposed into a small number of cells that correspond to rooms, doors, corridors, etc. They used a multisensory system to solve the inaccuracies introduced by the sensors; the approch was based on selecting the data collected by the more accurate sensor in a given situation; the data was selected based on averaging of different but consistent measurements of the same entity weighted with their associated uncertainties. Thrun and Buecken [56, 54] integrated grid-based and topological maps for the purposes of navigation. An integration of artificial neural networks with a Bayesian algorithm was implemented to build the grid map; topological maps were generated on top of the grid-based maps.

2.1.4 Comparing Geometric, Topological and Hybrid Approaches

All three environmental representations have some advantages as well as some shortcomings. Geometric maps require a large memory space, they are also computationally expensive. On the other hand, Topological maps require less memory space in comparison with the geometric maps. Also the computational cost of path planning can be reduced by using topological maps [54]. As the robot performs navigation from one node to another, there is no possibility of accumulation of global error,

which is a problem in geometric maps. However, topological mapping may become inefficient in case of environments with periodical structures [54], for example, an office environment where all the doors in a corridor are similar in color and texture and are regularly spaced; then place recognition by using a topological map becomes very difficult.

Topological maps in robotics may have some relation to human navigation. In order to go from one specific location to another, humans do not necessarily need to remember the goal location according to any precise metric information. It appears easier for humans to remember a place according to some recognizable scenes (distinctive landmarks) where specific actions are performed, such as turning right from a landing, entering a door, etc. Topological maps are constructed using nodes corresponding to significant places, and edges between them. Thus, this type of mapping may be easier for humans to use in comparison with the geometric maps. One major dissimilarity between metric map and topological map is that the robot's position must be detected accurately in metric map; but accurate metric position is not required for topological mapping. Hybrid representations contain the advantages of both paradigms. However, since metric mapping is based on local odometry information, so in case of erroneous data, the whole hybrid mapping system can become erroneous as the topological map is usually built on top of the metric map. Moreover, a large amount of memory space is required for hybrid maps, because both the metric and topological map are being built; even if the metric map is deleted after building the topological map, memory requirement is still large as both maps may be present together for sometime.

In our system, the environment is represented with a topological map. Motivation

for choosing such a representation mainly came from the technique humans use to remember a place. Tolman [60] first introduced the idea that internally humans represent the world environment as a *cognitive map*. Cognitive map can be defined as the spatial knowledge built up by observations acquired by a human being while traveling around in the world environment [34]. In our approach, we are less concerned with the geometry of the environment, rather the main importance is given to the fact whether the robot can reach a certain goal node in the environment from a different node; i.e. the geometric locations of the two places or nodes are not important in this case. In our system, the coordinates of a node are not stored in the map as they are not required. Each node in the topological map is represented with an image of that location. Any node in the topological map can be used as a start position or a goal destination for the navigation process.

2.2 Visual Homing

Visual homing can be defined as the method to reach a previously visited location using the stored image taken from that location. In order to return to the goal location, some kind of matching is performed between the current view image and the goal image. It can be observed that the current view image and the goal image are the basic requirements of a visual homing mechanism. The primary motivation of researches in visual homing of mobile robots came from the navigation behaviors in animals and insects. Biologists have done a lot of research works about visual homing techniques of insect navigation. The snapshot model of Cartwright and Collett [7, 8] is a well-known model for visual homing in bees. They developed the snapshot model

in order to detect how bees use landmarks to return back to their food sources. Collett [13] suggested that bees recognize places by performing some kind of image matching; they are able to return to a previously visited goal location by matching the current view as seen by them, with the stored representation of the goal location. According to [13], bees try to match features such as position and orientation of edges, their speed of motion, their color, etc.

Franz *et al.* [21, 22] developed a scene-based homing method which is one of the most highly cited methods for visual homing. In their method, the home vector is computed from the whole image, one-dimensional panoramic images were used. In order to find the home direction toward a goal location, the current view image is warped according to three parameters - the direction in which the robot has moved away from the goal, the change in the sensor orientation, and the ratio between the distance from the goal and the average landmark distance. The warped image can be thought of as the predicted goal image based on these three parameters. They constructed a matched filter that predicts the displacement field of landmarks; the selection criteria for the filter is based on the minimum image distance between the current view image and the goal image; the direction was calculated based on that specific matched filter. Relative to the robot's current position, these parameters specify a supposed goal position. The home direction is calculated by searching through the parameter space for the warped image that best matches the stored goal image. Their method includes orientation in its search space and therefore has the advantage of not requiring a compass. However, their assumption was that all the landmarks were approximately at equal distance from the location of the goal image. Most of the time, this assumption will be violated in natural environments, this makes their method

limited. Moreover, most of the works were tested in simulations; an experiment was done with a mobile robot using an artificial toy house environment; thus their method was not tested in natural environment. Zeil *et al.* [68] developed a view-based homing strategy using simple gradient descent methods. They investigated the behavior of root mean square (RMS) pixel differences between a reference image and a database of images, with the image distance and they found out that the home direction can be computed from the root mean square (RMS) pixel differences. The authors claimed that their method performed well even with transient illumination changes. Details of their view-based homing strategy can be found in section 4.4.

Vardy and Möller [64] investigated the performance of different optical flow techniques for the purpose of visual homing. They implemented the block matching method, two simple variants of block matching - intensity and gradient matching, and two differential techniques for homing. In block matching, a block of pixels are taken from one image and the matching pixels are detected in the other image; in intensity matching, a single pixel is taken instead of a block of pixels and gradient matching searches for the best-matching gradient between the two images. The authors used omnidirectional images for the experiments, they tested their methods using three different indoor environments. Results were compared with Franz et al.'s [21] homing method; their flow-based methods performed better than the reference method. The authors claim that their methods are able to perform even with some incorrect feature correspondences. Their analysis showed that matching between low-frequency image features is sufficient for homing. Recently, Churchill and Vardy [12] developed an algorithm for finding home vector, their method is based on Scale Invariant Feature Transform (SIFT) approach. In order to find the home direction from

a current location towards a goal location, first landmark correspondences are computed between the current view image and the goal image using SIFT. Based on the change in scale parameter of these correspondence vectors, a region of contraction and a region of expansion are detected in the current view image. According to [12], the home direction is aligned with the center of the region of contraction. However, the assumption of their method is that the objects are distributed uniformly throughout the environment. Details of their homing strategy can be found in section 4.3.

2.3 Path Planning

In a navigation system, after the robot has built a map (or the map can be given to the robot), it can start performing a navigational task such as moving from an arbitrary start position to some goal position. But, a proper path planning scheme must be employed in order to assure an efficient navigation process. Path planning provides the robot with a course of actions to reach a certain goal location, given the current location. A number of different techniques exist in the literature to compute a path in order to reach a goal position. One of the most famous strategies of path planning is Dijkstra's shortest path algorithm [16]; this algorithm has been used in numerous robotics research works [24, 35, 32]. Dijkstra's algorithm must be applied on a graph. It can be used on geometric maps also, but requires the space to be discretized. In this algorithm, a path from a source vertex to a target vertex is said to be the shortest path if its total cost (or some other measure can be used) is minimum among all the paths. The assumptions are that all the edge costs are non-negative. In this algorithm, the source must be a single vertex, but the target

can be all other vertices. The edges can be weighted or unweighted depending on the preference of the user.

Mataric [40] used a simple breadth-first search for path planning. Each landmark was given a weight according to its physical length, thus ensuring the computation of physically shortest path in the graph. In this method, a call propagates from the goal node in all directions of the graph and finds the current node, then the length of all the landmarks from the goal node to the current node is summed up, in this way the shortest path is computed. Lumelsky and Stepanov [39] proposed a different strategy for path planning where the shape and locations of the obstacles were not known by the mobile agent, in other words, no map was used for path planning. The robot would acquire the local information on its immediate surroundings from a sensor; its current location and the goal location would also be given. By designing a method for nonheuristic motion planning, they showed that this information was enough to reach the goal location.

Donnart and Meyer [17] presented a hierarchical classifier system for path planning and navigation. In order to reach a certain goal, the system plans a path using both both reactive and planning rules; *salient states* are defined in the path which can be referred to as intermediate goal locations, thus the robot reaches the goal while avoiding obstacles. Their system was tested with a simulated and a real robot.

2.4 Topological Navigation

Different strategies of topological map building were reviewed in section 2.1.2, here in this section, we will review different navigation methods in mobile robotics that use topological maps. Topological navigation can be defined as the process of going from one node to another goal node in the topological map. Crowley [14] was among the earliest successful researchers to develop a navigation system using topological representation. The author constructed a dynamically maintained model of the local environment called *the composite local model.* When the robot moved in the environment, the model integrated information from the rotating range sensor, the robot's touch sensor and a pre-learned global model. A network of places contained in the model, was used in path planning and navigation processes. Mataric [40] developed a topological navigation system on a reactive, subsumption-based mobile robot named *Toto* that was equipped with a ring of sonar sensors and a compass. Ulrich and Nourbakhsh [62] came up with an appearance-based place recognition system using an omnidirectional visual sensor. Their image classifier was based on nearest-neighbor learning algorithm, image histogram matching and a simple voting scheme. Their system was tested in four different environments, including indoor and outdoor; however, their method was not implemented and tested on a real mobile agent.

Fu *et al.* [23] implemented a passive mapper strategy that used sensor readings to create a topological representation of the world consisting of distinctive places and connecting edges. Their assumption was that the environment was structured

with known organization, but unknown specifications. However, they only tested their system in simulations. Kosaka and Pan developed two different methods for vision-based robot navigation in indoor environments. Their first approach named *FINALE* is based on a vision-based metric map and Kalman filtering. The second approach named *FUZZY-NAV* was based on a strategy integrating fuzzy logic and neural networks; topological representation of the environment was implemented for this approach. They used fuzzy logic to deal with the uncertainty in the visual data; the captured images were first processed by the neural network. The authors claimed that their system was able to navigate while avoiding both static and dynamic obstacles.

Goedeme *et al.* [24] developed a robot navigation system using topological representation. An omnidirectional vision system was used in their approach. A probabilistic approach was taken in order to perform image-based localization. Thus the location of the robot was obtained and using path planning methods, a path was computed to reach a certain goal position; the path was defined by a set of prototype images of places. To move the robot from one node towards the next node of the computed path, correspondences between the current image and the next node image were obtained; then using epipolar geometry, a homing vector was computed.

Chapter 3

Image Matching

3.1 Introduction

A sound image comparison technique is an essential part of a vision-based exploration and navigation system. Image matching techniques can be broadly categorized into two areas:

1. Global image comparison techniques

2. Local image comparison techniques

In global image comparison techniques, characteristics of the whole image are collectively used to describe a view. On the other hand, local image comparison techniques identify visually salient features in the image. The primary advantages of global techniques over feature-based techniques are that global techniques are simple and computationally fast; but they may perform poorly in the presence of occlusions. On the other hand, local techniques can be made robust against occlusions. Time complexity is a major disadvantage of local techniques.

3.2 Related Work

3.2.1 Global Image Comparison Techniques

Ulrich and Nourbakhsh used histogram-based matching technique for the purpose of place recognition for topological localization in [62]. Histogram-based matching is a well known method among the global techniques of image matching; advantages of histograms include less memory consumption, reduced computational effort (so that real-time experiments are possible with histograms), and invariance to image rotation. Ulrich and Nourbakhsh [62] used color panoramic images for their experiments; they argue that different places in an environment can easily be distinguished by their color appearance, which reduces the requirement of using range data from additional sensors such as stereo, sonar or a laser rangefinder for distinguishing different places. They used histogram matching to recognize the current location in the environment by comparing the current view image with the stored images taken at the currently believed location and its immediate neighbors. They also developed an adjacency relationship scheme that limits the number of reference images used in the comparison, thus their method does not need to compare the current view image with all the images in the database. But the main flaw of their method is that a single wrong matching can lead to an incorrect localization i.e. the robot will believe its in a certain location in the environment, but in reality it is in some other place. The authors performed eight cross-sequence tests in four unmodified environments, including both indoor and outdoor environments.

Aihara *et al.* [1] used eigenspace methods for image-based localization. Auto-correlation images (see figure 3.1), invariant against the rotation of the sensor, were

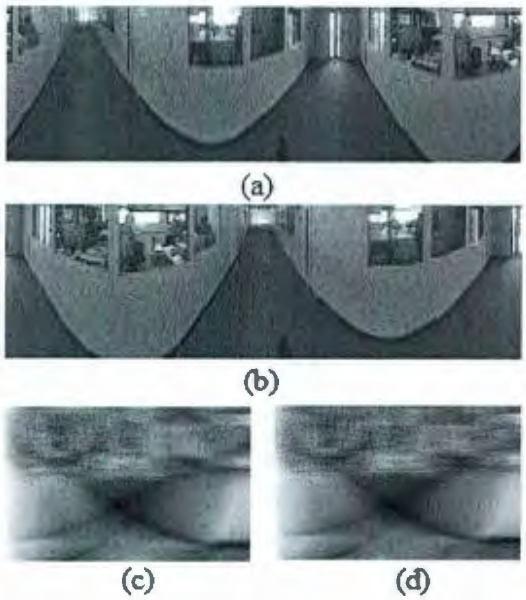

Figure 3.1: (a) and (b): Omnidirectional panoramic images taken at the same location with rotating the sensor 120-degrees, (c) and (d): Autocorrelation images transformed from (a) and (b), respectively. Image courtesy: Aihara *et al.* [1].

generated from omnidirectional images and the similarity of autocorrelation images was evaluated in low dimensional eigenspace. The authors claimed that their method could perform localization even with low dimensional images. However, their system was not implemented on a real mobile robot. Jogan and Leonardis [28] also used eigenspace decomposition technique for the purpose of spatial localization. They implemented the approach known as Zero Phase Representation (ZPR) which was proposed by Pajdla and Hlavác [44]. It provides a representation where many identical, but randomly oriented images, have the same ZPR, they referred to it as the solution of *one-to-many mapping problem.*

Menegatti *et al.* [41] used Fourier transforms for image-based localization. In their

method, each omnidirectional image was represented by the Fourier coefficients of the low frequency components of its panoramic conversion. Image similarity between two images was calculated using the the L1 norm of the Fourier measure of the images. However, all the test were done in simulations, mobile robots were not used to carry out their experiments. Stricker *et al.* proposed a method based on the Fourier-Mellin transform to compare images in [53]. Yagi *et al.* [67] also used a Fourier-based method for developing a route navigation scheme for a mobile robot; omnidirectional image sensor was used in their work. Omnidirectional images were represented by a series of two dimensional Fourier power spectra; image similarity was found by comparing the principal axis of inertia of the current position of the robot with that of the memorized Fourier power spectra. The assumption of their method was that the robot motion was constant and linear.

3.2.2 Local Image Comparison Techniques

There has been a great deal of work on local image comparison techniques. Local techniques are based on the detection of local features such as corners, doors, landmarks, certain types of artificial markers specified by the author, etc. Harris and Stephens [27] developed an algorithm to detect corners and edges in an image, which is a widely used algorithm for local feature detection in the field of robotics. Their algorithm is based on the local autocorrelation function, it detects the locations where the signal changes quickly in one direction (an edge), or in all directions (a corner). Schmid and Mohr [49] used the method of detecting local gray value invariants which was proposed by Koenderink and Doorn [29]. Interest points are automatically de-

tected from the image using the Harris corner detector [27], then differential invariants are calculated. A multiscale approach is used in order to obtain robustness against scale changes. They performed experiments with different conditions (such as - partial visibility, extraneous features, image rotation and scaling, and small perspective deformations) to demonstrate the robustness of their method. In all of these cases, their method was able to retrieve images properly from a database of more than 1,000 images.

Baumberg [4] proposed a scheme for detecting features to cope with local affine image transformations. In other words, they tried to detect the same features in identical images that are taken from different viewpoints. Using a multi-scale Harris feature detector [27], the interest points were first determined; then each interest point was characterized using affine texture invariants. These descriptors are calculated by normalizing for photometric intensity changes and removing stretch, skew and rotation effects. However, their method is not computationally efficient. Recently, Lowe [38] proposed the Scale Invariant Feature Transform (SIFT) approach for detecting features invariant to image scale and rotation. The author claims that the features are partially invariant to illumination changes and affine distortions and robust against changes in 3D viewpoints for non-planar surfaces. In this method, interest points invariant to scale and orientation were detected using difference-of-Gaussian function; then keypoint descriptors were generated containing location, scale and orientation information. Descriptors over a wide range of scale are detected in SIFT algorithm. Thus small and highly occluded objects can be identified using small local features; on the other hand, large features can be used to identify the objects from images distorted by noise.

3.3 Implemented Global Image Comparison Techniques

While exploring the environment, the robot may arrive at the same location or at a nearby location where another node was created previously. But we do not want to create a node at or very close to the same position where another node was created previously. So there must be some kind of mechanism in our system to recognize a previously visited place. Global image matching techniques can be used for this purpose. The method compares the current view image taken by the robot, with all the stored node images; then gives the decision whether a node should be created or not at the current location. In this section, we will describe all the global image comparison techniques which have been implemented in this thesis.

A sound image comparison technique is very important for an image-based exploration system. The aim is to determine a dissimilarity measure for each pair of images, which gives a measure of how visually analogous the two images are. We have used two global techniques for image comparison:

- Histogram-based techniques
- Fourier transform

The images used for the image comparison method are captured using an omnidirectional camera system. The most important advantage of the omnidirectional camera over a normal perspective camera is rotational invariance i.e. the same area of the environment is captured independent of the camera's orientation. Another

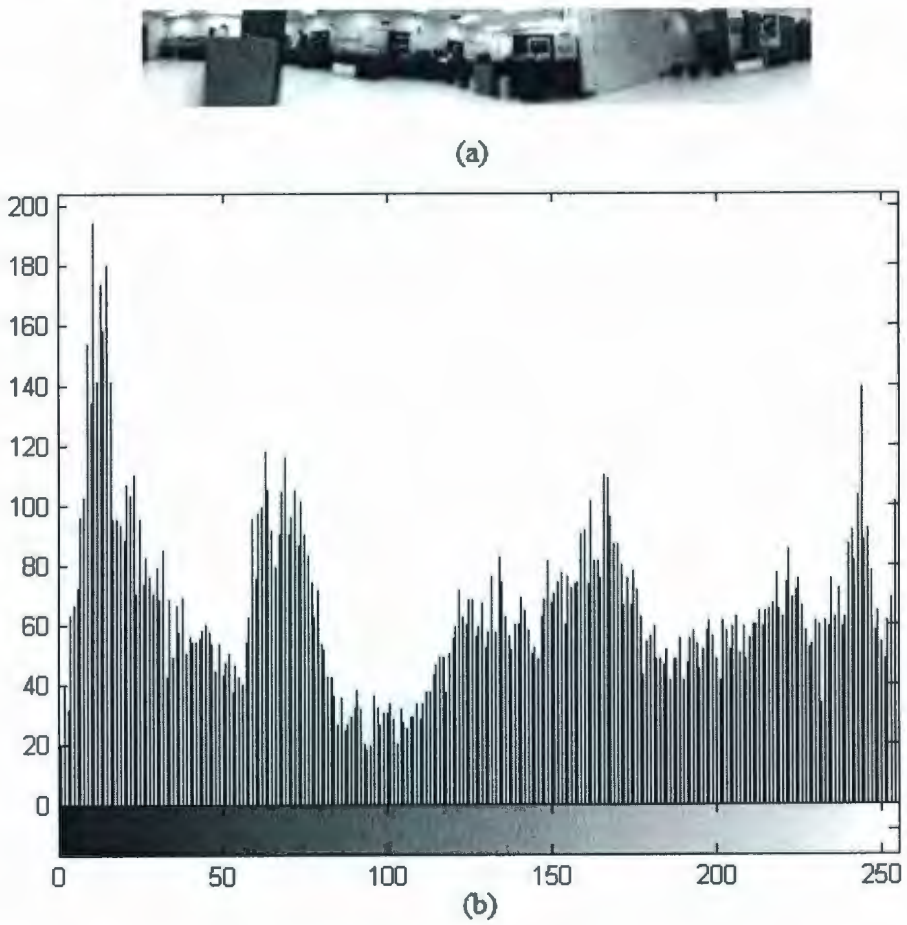

Figure 3.2: (a) A sample image, (b) Image histogram of figure (a).

advantage is the large field of view; this also makes the system robust against small changes in the environment.

3.3.1 Histogram-based techniques

For an image-based localization system, one of the major problems is to store a large number of images in the memory database, which takes a large amount of space. Histograms are good solutions for this problem; they require very little memory

space. In our experiments, a color omnidirectional image requires approximately 2.25 MB memory space, a gray level panoramic image requires approximately 20 KB memory space, whereas the image histogram of the gray level panoramic image requires approximately 1 KB memory space. As a result, the amount of memory required to store an image can be reduced. The image histogram of an omnidirectional image is rotationally invariant. In other words, if two omnidirectional images are captured at a certain location of the environment with different orientations, then the image histograms of these two images will be identical. Figure 3.2 shows a sample image and its image histogram.

In order to determine how well two image histograms match, three histogram matching techniques have been used in this research work, they are given below:

- Jeffrey divergence method
- χ^2 statistics method
- Sum of absolute difference method

A good overview of different histogram matching techniques is given in [47]. The Jeffrey divergence method is numerically stable and robust with respect to size of histogram bins [46]. The dissimilarity measure in this method is defined as:

$$d_J(H_i, H_j) = \sum_k (h_{ik} \log \frac{h_{ik}}{m_k} + h_{jk} \log \frac{h_{jk}}{m_k}) \tag{3.1}$$

where, $m_k = \frac{h_{ik}+h_{jk}}{2}$ and h_{ik} and h_{jk} are the histogram entries of the two image histograms H_i and H_j respectively.

The χ^2 statistics method was also used in [62]. The dissimilarity measure in this method is defined as:

$$d_{\chi^2}(H_i, H_j) = \sum_k \frac{(h_{ik} - m_k)^2}{m_k} \tag{3.2}$$

where again, $m_k = \frac{h_{ik}+h_{jk}}{2}$ and h_{ik} and h_{jk} are the histogram entries of the two image histograms H_i and H_j respectively.

The last image comparison method is a straightforward one, the dissimilarity measure is obtained by the sum of the absolute differences of the two image histogram entries:

$$d_S(H_i, H_j) = \sum_k |h_{ik} - h_{jk}| \tag{3.3}$$

where, h_{ik} and h_{jk} are the histogram entries of the two image histograms H_i and H_j respectively.

The following example gives an overview on the three histogram methods used in this work: a reference histogram is shown in figure 3.3 and two other histograms are given in figure 3.4. The objective is to calculate the dissimilarity measure measure between the reference histogram and other histograms. The histogram entries are {6, 26, 37, 26, 3, 2}, {3, 25, 38, 25, 8, 1} and {34, 36, 4, 14, 10, 2} for the reference histogram, histogram1 and histogram2 respectively. It can be observed from figures 3.3 and 3.4 that the resemblance between the reference histogram and histogram1 is higher than the resemblance between the reference histogram and histogram2. As a result the dissimilarity measure between the reference histogram and histogram1 should be lower than the dissimilarity measure between the reference histogram and

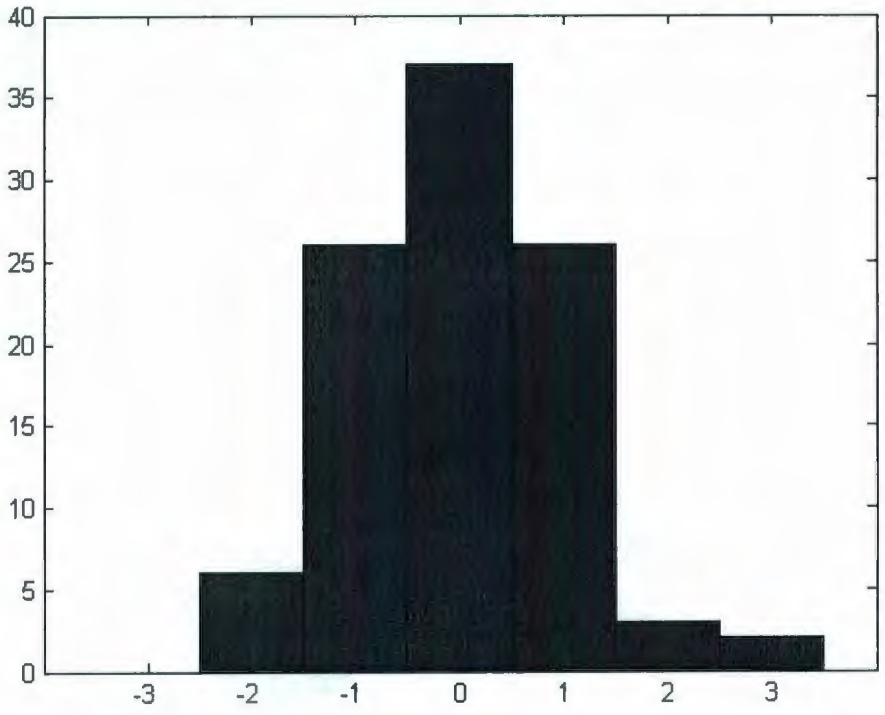

Figure 3.3: Reference histogram

histogram2. The dissimilarity measures found using equations 3.1, 3.2 and 3.3 also verify the above observations. The results are given in table 3.1.

3.3.2 Fourier transform

Similar to image histograms, the magnitude of the one-dimensional Fourier transform of the rows of an omnidirectional image is invariant to the rotation of the image around the optical axis. The panoramic image is transformed row by row via the Fourier transform.

$$d_F(I_i, I_j) = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} |F_{ik}(l) - F_{jk}(l)| \tag{3.4}$$

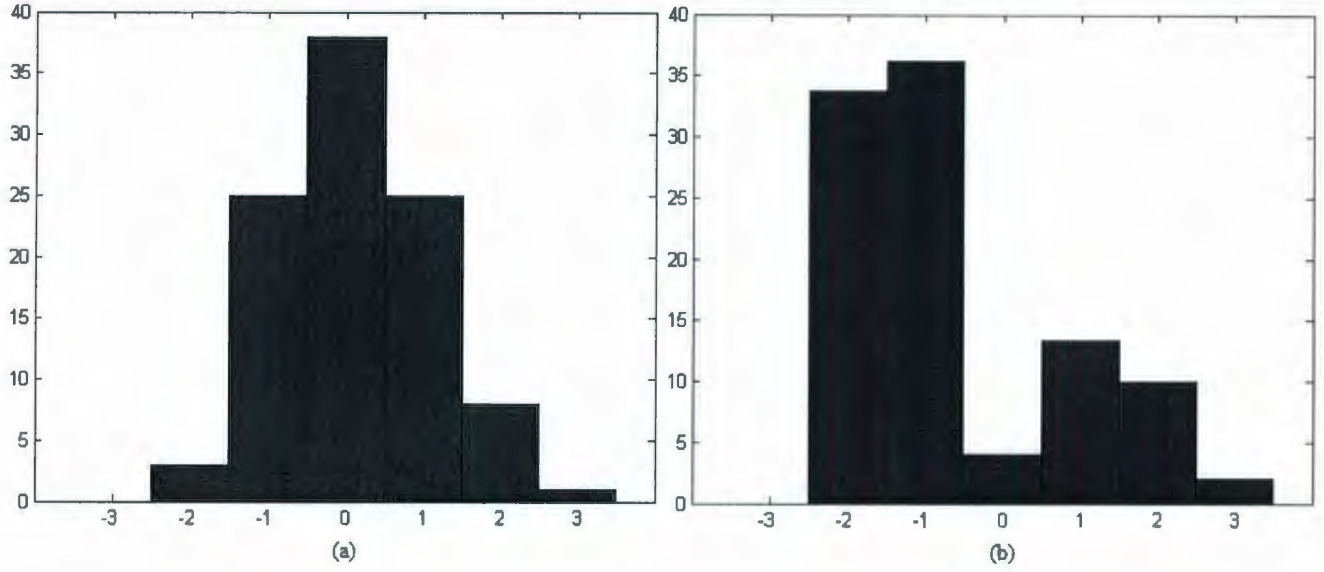

Figure 3.4: (a) Histogram1, (b) Histogram2.

Dissimilarity measure	Reference Histogram and Histogram1	Reference Histogram and Histogram2
Jeffrey divergence	0.8187	13.3570
χ^2 statistics	1.8293	27.5716
Sum of absolute difference	12	90

Table 3.1: Dissimilarity measures using the reference histogram, histogram1 and histogram2.

where, I_i and I_j are the two panoramic images, each having m rows. $F_{ik}(l)$ and $F_{jk}(l)$ are the Fourier coefficients of the l^{th} frequency of the k^{th} row of images I_i and I_j respectively. This method was also used in [41].

The Fourier coefficients of the low frequency components of the panoramic image are stored to represent the image. In our experiments, we took the first 30 frequency components, because the later frequency components have very small values and thus can be neglected in the calculation of the dissimilarity measure.

3.4 Image Based Localization

Image-based localization consists of matching the current view image experienced by the mobile robot with training images stored in the memory of the robot. In a new environment, the mobile robot is lead along a route and training images are captured. Then if a new test image is captured, it is compared with all the training images and an hypothesis is formed about the current location of the mobile robot. This hypothesis is refined using the discrete Bayes filter as soon as the robot starts to move and new test images are captured. So the output of image-based localization system is a location which refers to one of the training images.

In this work, our experiments were done for two types of localization problems, namely local localization and global localization. When a mobile robot first starts to localize, it has no knowledge of its location in the environment; this is known as global localization. In the case of local localization, the initial location of the robot is known by the mobile robot. Global localization is more difficult than local localization, because at the beginning there is no knowledge about the location of the robot, so the algorithm starts with equal probability given to each training image.

Our image-based localization system is able to perform both types of localization. A probabilistic approach is used in this work i.e. we will represent the robot's belief of its location as a probability distribution. There are a number of ways to represent probability distributions: continuous or discrete, single or multiple hypothesis. In this work, we used a discrete Bayes filter, with probability distribution approximated by an array of possible locations (i.e. training images).

3.4.1 Bayes filter

In the Bayes filter algorithm, the probability distributions are calculated from measurement and control data. Bayes filter utilizes the Markov assumption, or the complete state assumption. According to this assumption, the past and future data are independent if one knows the current state [58]. Bayes filter is recursive i.e. the robot's belief *bel(x_t)* at time t is calculated from the belief *bel(x_{t-1})* at time *t-1*; where, x_t is the robot's state at time t. The input for Bayes filter is the belief *bel(x_{t-1})* at time *t-1*, the most recent control input u_t and the most recent sensor measurement z_t ; the output of the algorithm is the robot's belief *bel(x_t)* at time t.

The general form of the Bayes filter is given below:

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1} \tag{3.5}$$

$$bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t) \tag{3.6}$$

where, $\overline{bel}(x_t)$ is the predicted belief and η is a normalizing factor. A belief distribution assigns a probability to each possible hypothesis with regards to the true state [58]. Belief distributions are posterior probabilities over state variables conditioned on the available data. It predicts the state at time t based on the previous state posterior, before incorporating the measurement at time t.

Equation 3.5 updates the belief to account for the robot's motion. This generates the **prediction** $\overline{bel}(x_t)$. Equation 3.6 achieves the **measurement update**. It incorporates the sensor values and combines this information with the prediction to update the belief.

In this work, we used a discrete Bayes filter which is given by equations 3.5 and 3.6, with the exception that the integration is replaced by summation. In order to localize the mobile robot, a probability distribution is maintained over all the nodes of the graph. So the output of the discrete Bayes filter is a probability distribution over all the nodes of the graph. The node with the highest probability value refers to the probable location of the mobile robot in the environment.

3.4.2 Motion Model

The motion model $p(x_t|u_t, x_{t-1})$ gives the probability of a transition from position x_{t-1} to x_t. Generally motion models are based on odometry information. The motion model of a differentially driven robot was used in [2].

If a mobile robot is at a certain location in the environment and it makes a forward motion, it is very probable that it will move to a neighbor location in the next time instant; the probability of moving to a place far from its current location is very low. As a result the motion model can be represented using a Gaussian probability distribution, as used in [24]. The motion model is defined as:

$$p(x_t|u_t, x_{t-1}) = \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{\frac{-dist(x_t, x_{t-1})}{\sigma_x^2}} \tag{3.7}$$

In the above equation, the function $dist(x_t, x_{t-1})$ refers to a measurement of the distance between the two places x_{t-1} and x_t; and σ_x is the standard deviation of the distances. In our experiment, the robot moves from the current node to the next one in each time instant, we assumed the distance between two adjacent places or nodes to be 1 unit. In our experiments, the robot can either move one step forward or stay

at the same place in case of no movement; thus a value of $\sigma_x = 1$ is reasonable.

3.4.3 Measurement Model

The measurement model $p(z_t|x_t)$ gives the probability of acquiring sensory observation z_t under the assumption that the robot is positioned at x_t.

In mobile robotics, different types of sensors are used to acquire sensory observations, such as laser range finders, ultrasonic sensors, camera, etc. The measurement model in [61] is composed of some discrete and continuous measurements: node degree, node equidistance, edge travel distance and feature map landmark location. Goedeme *et al.* [24] used a Gaussian probability distribution to represent the measurement model and we have adopted this approach. As mentioned in [24], there exists a low probability of acquiring an image at a certain location that differs substantially from the training image taken at that location. The measurement model is defined as:

$$p(z_t|x_t) = \frac{1}{\sqrt{2\pi\sigma_z^2}} e^{\frac{-diff(h_1,h_2)}{\sigma_z^2}} \tag{3.8}$$

In the above equation, the function $diff(h_1, h_2)$ is obtained by image comparison methods described in section 3.3, and σ_z is the standard deviation measured on the data. In our experiments, the value of σ_z is obtained using the current test image and the current sequence of training images. For example, for each test image and the training images of sequence *original_1*, first the dissimilarity values between the test image and each training image of the sequence are obtained using the image comparison techniques described in section 3.3. Then the standard deviation σ_z is

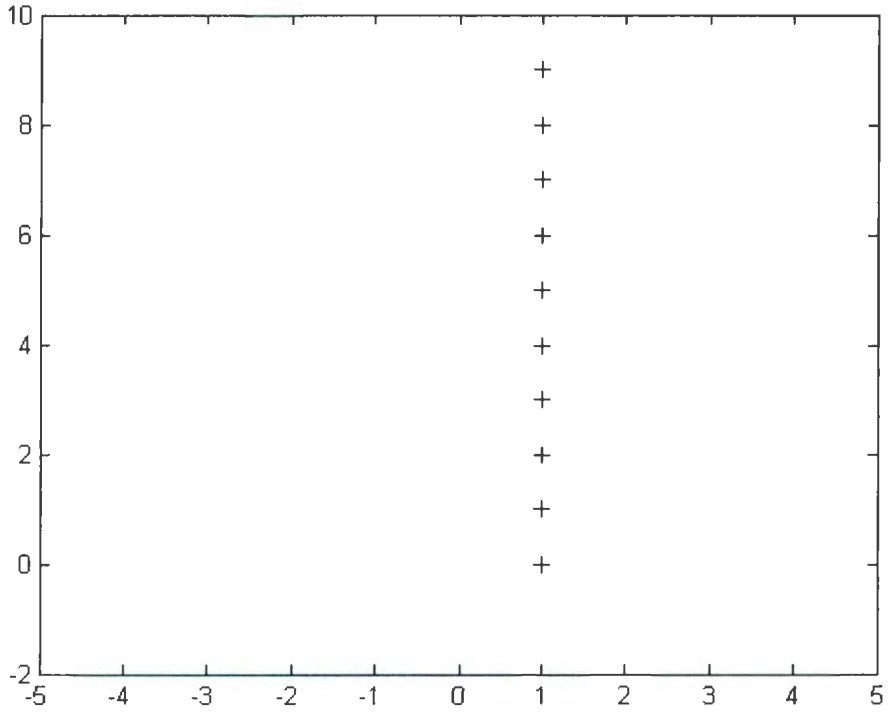

Figure 3.5: Experimental image setup for image comparison with all four global techniques. Black plus sign at $(1, 0)$ refer to the test image and all the 10 black plus signs refer to the training images (30 cm apart). Images are from the image sequence *original_1*.

calculated from these dissimilarity values.

3.5 Image Comparison Experiments and Results

3.5.1 Image database

In our experiments, an image database was used which was created from the images captured in the robotics laboratory of Bielefeld University. This image database is

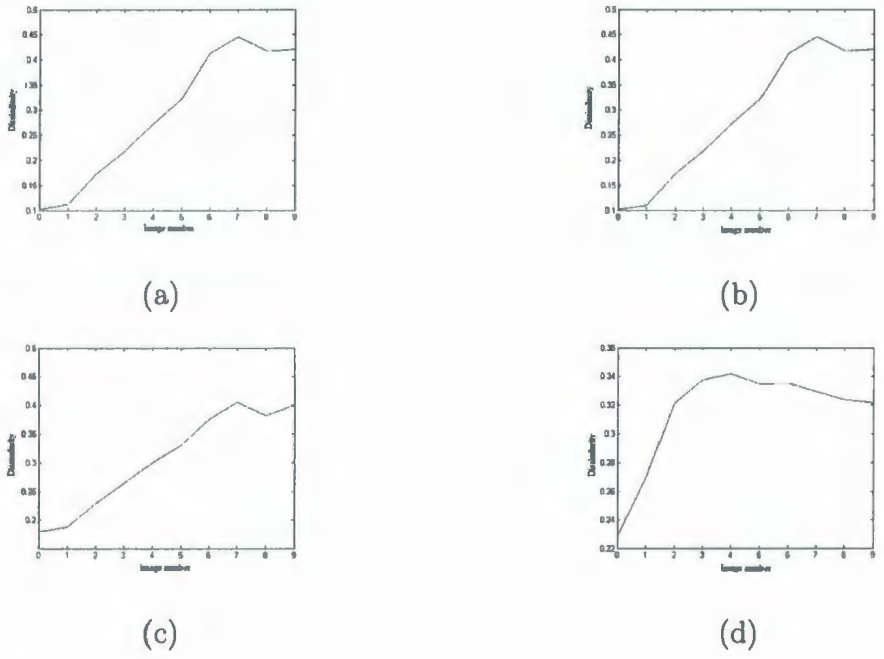

Figure 3.6: Image comparison with (a) Jeffrey divergence (b) χ^2 statistics (c) Sum of absolute difference method and (d) Fourier transform. The test image is the first image of the image sequence *original_1* positioned at (0,0) and the training images are the first 10 images from the image sequence *original_1*.

publicly available at *www.ti.uni-bielefeld.de/html/research/avardy*. Images were collected by a camera mounted on a pioneer mobile robot. The camera was a catadioptric system consisting of an upward looking camera with a hyperbolic mirror mounted over it. The hyperbolic mirror expanded the camera's field of view to allow the capture of omnidirectional images. A detailed description of the image databases and the catadioptric vision system can be found in [64].

Figure 3.6 shows the dissimilarity measure using both histogram and Fourier transform methods. The image setup for all the image comparison methods are shown in figure 3.5. Here, the black plus sign at $(1, 0)$ refer to the test image and all the 10

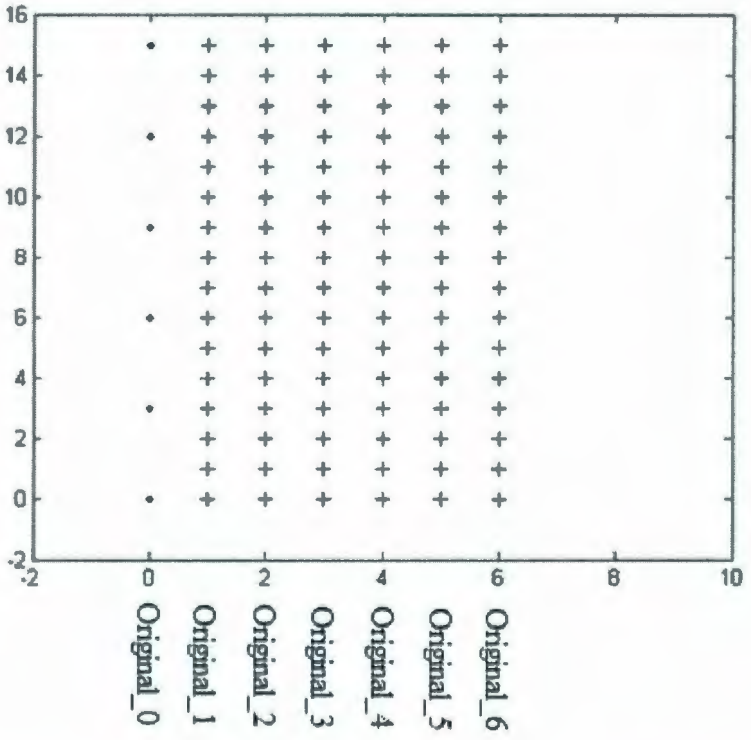

Figure 3.7: Experimental image setup for image-based localization. Black dots refer to test images (90 cm apart) and black plus signs refer to training images (30 cm apart). The image sequences are 30 cm apart from each other.

black plus signs refer to the training images (30 cm apart). Images are from the image sequence *original_1*. The dissimilarity function behaves as expected in all four cases. The dissimilarity function is 0 when the test image and the training image are the same (for image position $(1, 0)$) as can be seen from figure 3.6, then the value of the dissimilarity function increases for other images. In other words, the dissimilarity function increases with spatial distance and after reaching a certain distance it will saturate. The reason of such behavior is that in case of two images taken from completely different place in the environment, there is no correlation at all between the

two images.

3.5.2 Experiment-1: With distant test and training images

We want to compare the performance of the four global image matching techniques in order to find the one that is best to use in our online exploration and navigation system. For this reason, we performed two experiments using the image database stated before. Figure 3.7 shows the experimental image setup for image-based localization system. Black dots refer to training images (90 cm apart) and black plus signs refer to test images (30 cm apart). There are 6 images in the sequence of training images and 16 images in the sequence of test images, as can be seen from figure 3.7. The image sequences are 30 cm apart from each other.

Our image-based localization system is able to perform both local and global localization. Global localization is performed by initializing the system with uniform probability distribution; while for local localization, the initial location of the mobile robot was given. The task is to determine which training image the robot is closer to for a certain test image. If, for example, the location where the test image is captured is closer to the second training image, then the second training image should have the highest probability value. Figure 3.8 demonstrates image-based global localization. Black dots refer to training images (90 cm apart) and the black plus sign refers to the current image (i.e. test image). In figure 3.8(a), the system is initialized with uniform probability distribution, so each training image has the same probability value (depicted by equal sized circles around the black dots). When the robot moves forward, the first training image obtains the highest probability value, as this training

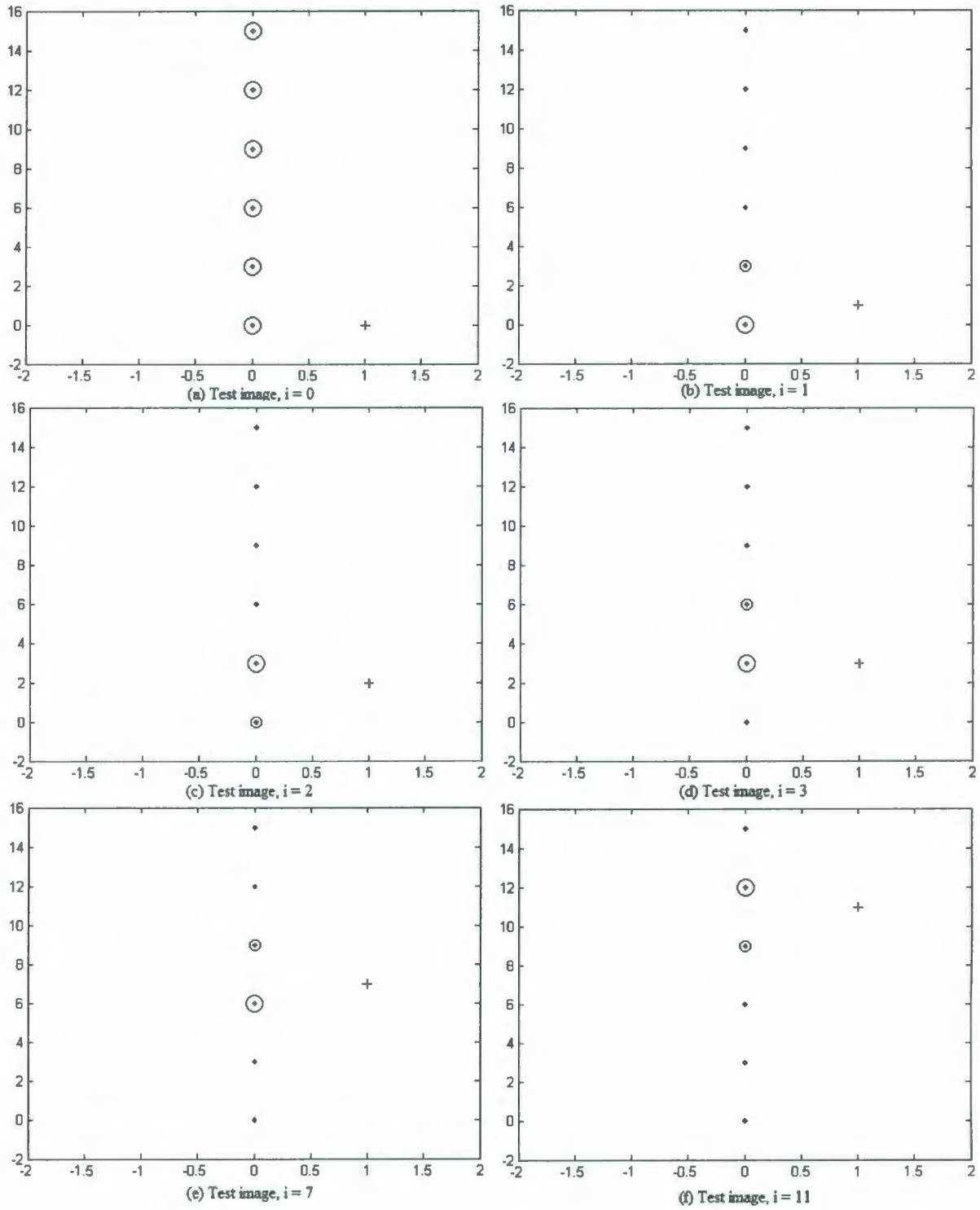

Figure 3.8: Demonstration of image-based global localization.

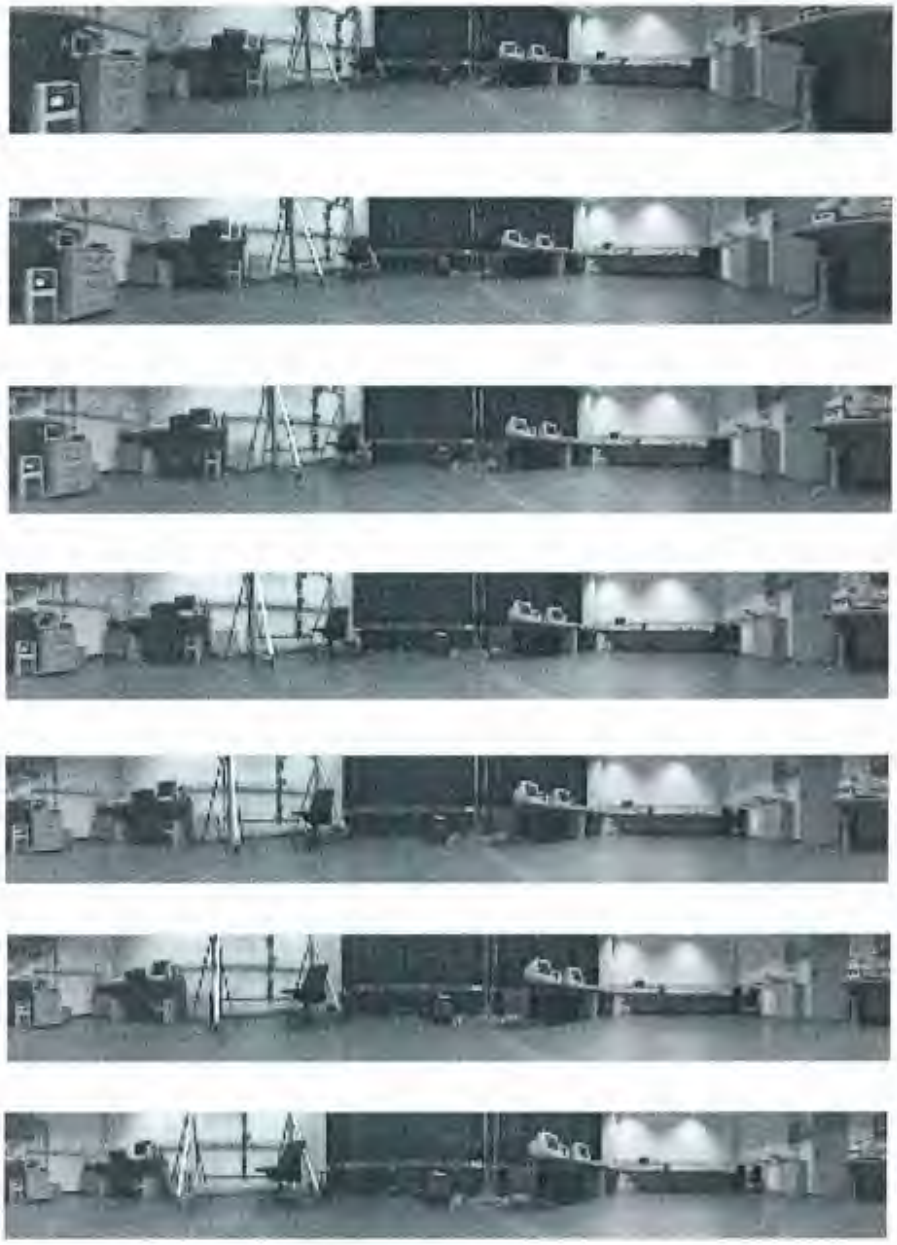

Figure 3.9: Sample images from image database *original_1*, images are 30cm apart. The distance between the sequence of test images and that of training images is 0, 30, 60, 90, 120, 150 and 180 cm respectively.

Test image database	Training image database	Jeffrey divergence	χ^2 statistics	Sum of difference	Fourier Transform
Original_0	Original_0	100%	100%	100%	100%
Original_0	Original_1	100%	100%	100%	100%
Original_0	Original_2	93.75%	81.25%	75%	81.25%
Original_0	Original_3	81.25%	75%	56.25%	75%
Original_0	Original_4	75%	56.25%	37.5%	62.5%
Original_0	Original_5	25%	18.75%	18.75%	56.25%
Original_0	Original_6	12.5%	12.5%	12.5%	31.25%

Table 3.2: Image-based Mobile Robot Localization Results (With distant test and training images)

image is the closest to the current test image; the training image with the highest probability value is depicted by a large circle and the training image with the second highest probability value is depicted by a small circle.

Figure 3.9 shows the sample images used in this experiment. The images are from the image database *original_1*. For the first experiment, the distances between the sequence of training images and that of test images were kept at 0, 30, 60, 90, 120, 150 and 180 cm respectively. The comparison is based on the ratio of successful localizations; for example, if the number of test images is 16 and the number of successful localization is 15, then the result is 93.75%.

The results of experiment-1 are summarized in table 3.2. It can be observed from the table that for all four methods, the ratio of successful localizations was almost 80% even when there was 60 cm distance between the training and test images (for test database *original_0* and training database *original_2*); afterward, the ratio decreased as the distance between the training and test images was increased. The ratio of successful localizations was 100% for training databases *original_0* and *original_1* for all four methods. In this experiment, the Jeffrey divergence method performed

the best among all the methods; the ratio of successful localizations was 75% even when there was 120 cm distance between the training and test image databases (for test database *original_0* and training database *original_4*). The performance of the Fourier transform method was also satisfactory; the ratio of successful localizations decreased gradually in this method. It should be noted that the Fourier transform method performed better than the the Jeffrey divergence method in case of the highest distance between the training and test image databases (for test database *original_0* and training database *original_6*). The performance of the χ^2 statistics and the Sum of absolute difference method was good when the distance between the training and test image databases was lower; but they did not perform as well as the Jeffrey divergence and the Fourier transform method when the distance was increased.

3.5.3 Experiment-2: With modified environments and illumination changes

For the second experiment, the sequence of test images and that of training images was 30 cm apart; but the environment was modified, four different training image databases were used. There were also some illumination changes in different image databases. Fig. 3.10 shows sample images from the image databases: *original, night, twilight and winlit.* Image database *original* refers to the standard or default condition of the room, with the curtains and door closed. Images of the database *night* were captured at night with the curtains and door open. Images of the database *twilight* were captured just after the sunset, at that time the room was still receiving plenty of daylight, the curtains and door were kept open. In the images of the database *winlit*,

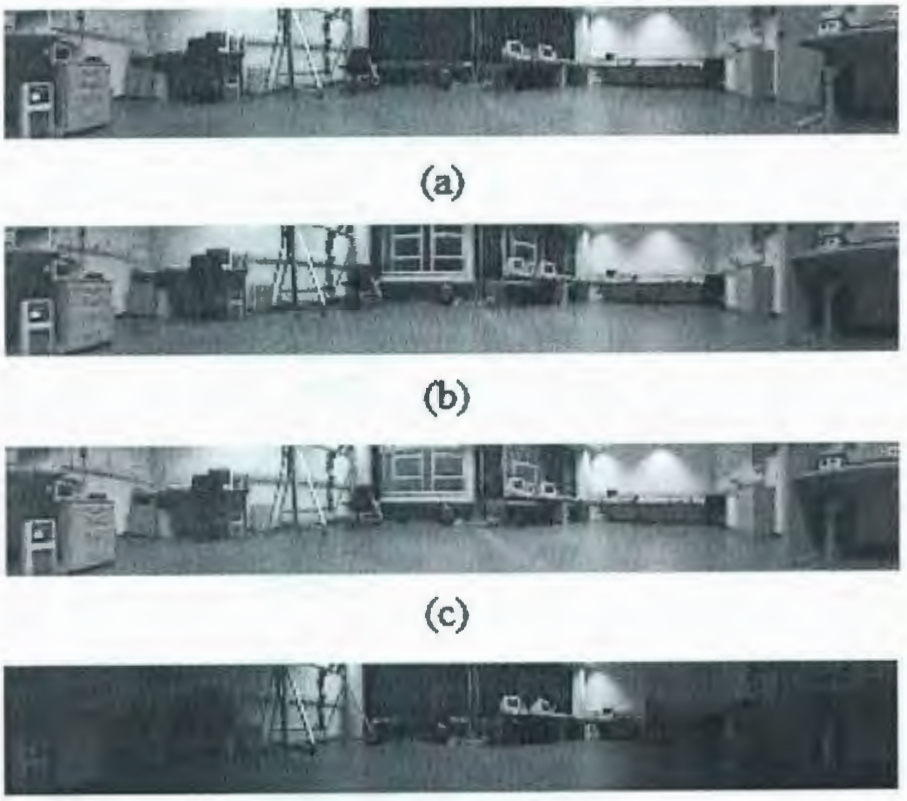

Figure 3.10: Sample images from 4 image databases: (a) *Original*, (b) *Night*, (c) *Twilight*, (d) *Winlit*. Image position (1,2).

only the two lights near the window were kept on.

The results of this experiment are summarized in table 3.3. It can be observed from the table that our image-based localization system was able to perform well even with modified environments and illumination changes. The ratio of successful localizations was 100% for training image database *original_1* for all the four methods. The ratio of successful localizations was above 75% for the training databases *night_1* and *twilight_1*; although it was above 93.75% for the Jeffrey divergence and the Fourier

Test image database	Training image database	Jeffrey divergence	χ^2 statistics	Sum of difference	Fourier Transform
Original_0	Original_1	100%	100%	100%	100%
Original_0	Night_1	93.75%	87.5%	81.25%	100%
Original_0	Twilight_1	93.75%	81.25%	75%	93.75%
Original_0	Winlit_1	18.75%	6.25%	6.25%	37.5%

Table 3.3: Image-based Mobile Robot Localization Results (With modified environments and illumination changes.)

transform method. It should be noted that no method performed well when the test images were taken from the image database *winlit_1*; because only two lights in the room were kept on for the images in the database *winlit*, so the room was really dark as can be seen from figure 3.10(d). This image database has significant illumination change from all the other databases, we intend to look further into this illumination change problem in our future works.

It can be observed from the tables 3.2 and 3.3, that both the Jeffrey divergence method and the Fourier transform method performed well in our experiments; we chose the Jeffrey divergence method to use in the image comparison of our online exploration and navigation system. The main reason for our choice was that the topological map was built in real time on-board the mobile robot in our system, so we wanted an image matching technique that was robust as well as computationally fast. The Jeffrey divergence method is simpler and presumably more efficient. We performed a simple test to test the running time of all four methods. We took two images and computed the dissimilarity value using all the four methods and recorded the time taken by each method. The recorded times were 0.014 sec, 0.006 sec, 0.005 sec and 0.034 sec for Jeffrey divergence method, χ^2 statistics method, sum

of absolute difference method and Fourier transform method respectively. Although the χ^2 statistics method and sum of absolute difference method took less time to compute the result, our experimental results from the tables 3.2 and 3.3 show that the performance of Jeffrey divergence method Fourier transform method are better than them. As the Jeffrey divergence method took less time to compute the result than the Fourier transform method, we chose the Jeffrey divergence method for the purpose of image matching in our online exploration and navigation system.

Chapter 4

System Architecture

4.1 Introduction

Our system uses an omnidirectional camera system and a laser range finder for performing the exploration and navigation task properly; although the navigation process relies only on the omnidirectional camera. The overall system architecture is shown in figure 4.1. Specially made artificial environment is not a requirement for our system which is able to perform exploration and navigation in natural environments. As stated before, a topological approach has been used to map the environment. In our system, a node is represented by an omnidirectional image captured at that location of the environment; no geometric information is used to represent the nodes in our system. When the robot is in a new environment, it starts to explore the environment. The laser is used for finding the best exploration direction; for each node, the best four exploration directions are stored. Our exploration algorithm is described in section 4.2. The robot turns according to the best exploration direction and then

moves forward. The robot keeps moving until a certain distance is reached, then it checks if the current view is similar to any known node of the topological map; since we do not want to create a node at or very close to the same position where another node was created previously. Otherwise, two nodes will be created in the topological map which represent the same or nearby location in the world environment, thus making the topological map inefficient. To get rid of this problem, we have used an image matching technique. The method compares the current view with all the stored images corresponding to each node; then gives the decision whether a node should be created or not for the current position. So in our system, only visual information is used to check the similarity of the current view image with the node images. The comparison is done by histogram matching, details of the image matching technique can be found in chapter 3.

If the current view is not similar to any known node, then a new node is created; an image is captured and stored to this new node and the 360^0 laser data is also obtained for that node. In our system, an edge is automatically created between the new node and the previous node. Then the robot should start exploration again for the new node, it chooses the best exploration direction obtained from the laser data as described before and moves forward. In this way the robot explores the whole environment.

On the other hand, if the current view is similar to any known node, then the robot starts homing back to that node; this ensures that a new node is not created on the same or near location of a previously crenated node. Homing is done using only the images stored at each node. The homing algorithm used in our system is described in section 4.3. After reaching the node an edge is created between the

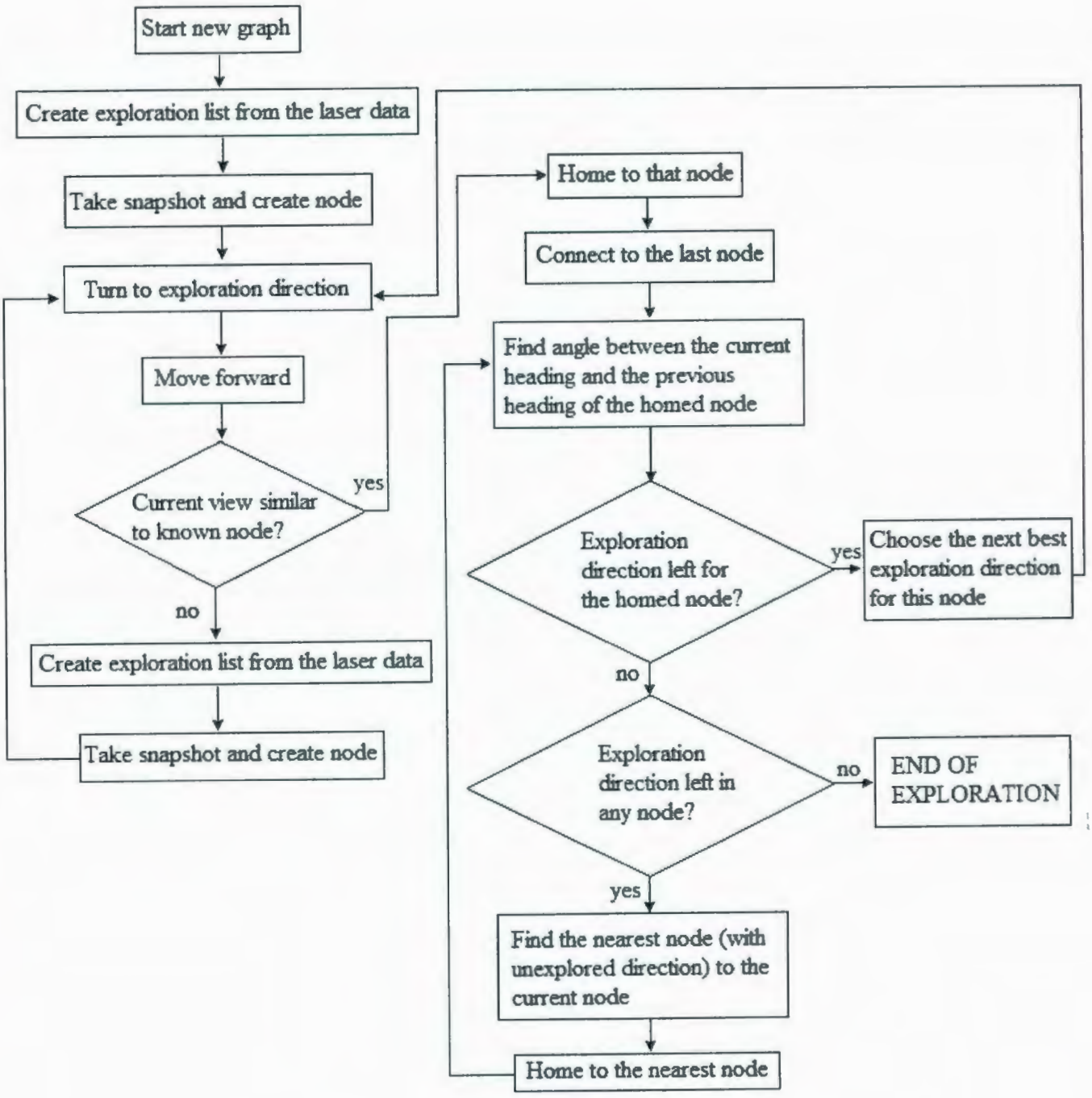

Figure 4.1: Whole System Architecture

previous node and the homed node; in this way nodes in the topological map are connected. This also serves the purpose of loop closing. Then the robot continues exploring the environment according to the remaining exploration directions of the homed node which were stored when the node was created in the topological map. But the exploration direction is based on the heading of the robot when the node was first created, not based on the current heading of the robot. For this reason, it is necessary to find the difference in angle between the current heading and the previous heading of the homed node. This is done using a visual compass algorithm based on gradient descent in image distances of the two images; the description of this algorithm can be found in section 4.4. So this algorithm gives the angle difference between the current heading and the heading of the robot when this node was created, the robot is turned according to this angle; then the robot takes the next best exploration direction remaining for this homed node, turns to that angle and starts exploration again. However, if there is no more exploration direction remaining for the homed node, then path planning is activated.

In path planning, first the topological map is checked to see if there is any exploration direction left for any node. If no exploration direction exists in the current topological map, then the robot has finished the exploration, so path planning state is stopped and the system is in the 'END OF EXPLORATION' state; now since the exploration is complete the robot can perform navigation tasks. On the other hand, if exploration directions still exist for any of the nodes, then the path planning state is continued. The nearest node (with remaining unexplored direction) is chosen using Dijkstra's shortest path algorithm. Then the robot continues exploring the environment according to the remaining exploration direction of the nearest node.

4.2 Finding Directions for Exploration

We need a strategy to explore the environment from any node of the topological map. In our system, a laser range finder is used to find the exploration directions. The laser range finder can sense objects in distances of up to 40 to 60 meters, depending on the object's reflectivity. The objective is to drive the robot towards the largest open space in the environment.

Outline of the Exploration Direction Algorithm:

- Filter laser data
- Find continuous open angle segments
- Sort the open angle segments
- Find the starting index of each continuous open angle segment
- Calculate the exploration directions for each continuous open angle segment

Figure 4.2 shows a 360^0 laser data plot, the data was taken in an office environment. As can be seen from the figure that sometimes few laser data can be erroneous; in our experiments, some distances were often found to be almost $32m$ while the lab was $10m \times 12m$. On the other hand, some recorded distances were very close to $0m$ which is not possible since the obstacle avoidance system will not let the robot to be so near to any obstacle. For this reason, the first step in our exploration direction algorithm is to filter the laser data, any data smaller than $10m$ was filtered out; similarly, any data greater than $15m$ was filtered out. Figure 4.3 shows a plot of the laser data of figure 4.2 after filtering.

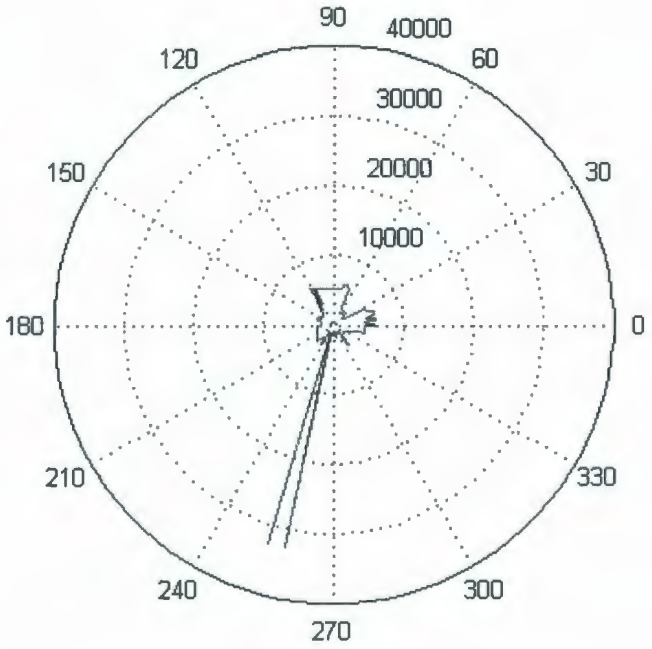

Figure 4.2: Plot from unmodified laser data

As can be seen from figure 4.3 that there can be a number of exploration directions from a node in the map. In our method, the exploration directions are calculated based on the largest open angle segments. In order to explain our exploration direction algorithm properly, we have used a sample laser data plot as can be seen from figure 4.4. It can be observed from this figure that there are three potential exploration directions for this particular location in the environment, which are shown by the three continuous open angle segments (shown by the blue arc). First, the maximum range data is obtained from the 360^0 laser data and then the continuous open angle segments are calculated for that particular set of laser data. Based on this maximum range data, an interval is defined to select the continuous open angle segments; all the laser data in each segment must be within this interval. It should be noted that

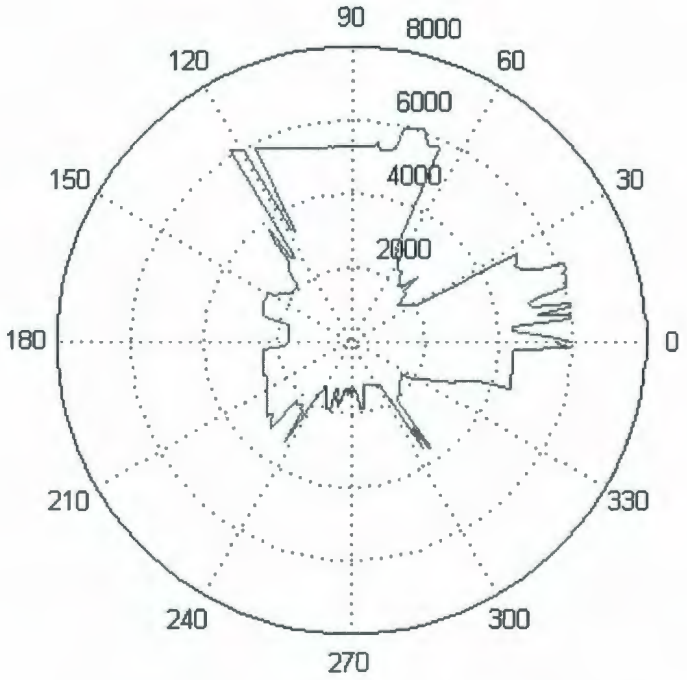

Figure 4.3: Plot from modified laser data

we are selecting open angle segments which are continuous, the reason is that the environment may contain some open places that has sharp edges or obstacles in the middle; in this case, the robot will not be able to move the required distance to distinguish the current view image from the previous node images. So the open angle segment must be continuous to ensure proper exploration.

We want to store the best three exploration directions at each node. For this reason, the continuous open angle segments are sorted in a descending order and the first three segments are taken for calculating the exploration directions. In our algorithm, the exploration directions are calculated in angles, these angles must be calculated with respect to some reference point in the laser data plot. For this reason, it is necessary to find the starting index of each continuous open angle segment; each

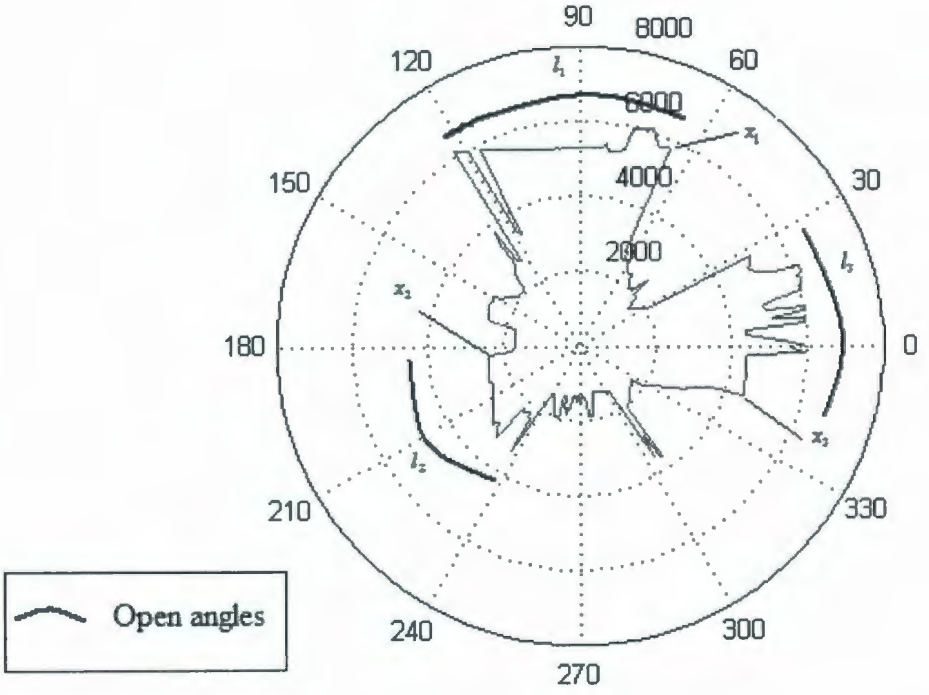

Figure 4.4: Modified laser data plot with continuous open angle segments shown.

starting index is calculated with respect to the reference point in the laser data plot. Then the exploration angle is calculated for each continuous open angle segment using equation 4.1:

$$\theta = x + \frac{l}{2} \tag{4.1}$$

where, θ is the exploration direction, x starting index of the continuous open angle segment and l is the length of the continuous open angle segment.

There can be some long narrow spaces in the environment which result in large distances from the robot, but in reality the space may be too narrow for the robot to explore that space. In order to get rid of this kind of misleading laser data, the algo-

rithm was modified such that if the continuous open angle segment was smaller than an experimentally determined threshold, then it would not be chosen as a potential exploration direction for that node.

4.3 Visual Homing

When the current view image is similar to any known node of the topological map, we want to move the robot to the similar node and continue exploration from there. This can be done by visual homing methods. Visual homing can be defined as the ability to return to a goal location by performing some kind of matching between the current view image and the goal image.

There have been quite a lot of research works on visual homing. We have used the homing algorithm developed by Churchill and Vardy [12]; their algorithm is based on the scale invariant feature transform (also known as SIFT) algorithm developed by Lowe [38]. Lowe [38] developed an algorithm to extract keypoints from an image which are invariant to image rotation and scale; and robust to a certain extent against affine distortion, addition of noise, change in 3D viewpoints and illumination changes. According to the author, the main advantage of these features is that they are highly distinctive, for this reason a single feature can be properly matched with high probability against a large database of features. The generation of SIFT feature points can be divided into four stages: scale-space extrema detection, keypoint localization, orientation assignment and keypoint descriptor. In the first stage, a difference-of-Gaussian function is applied to detect the scale-space extrema; the search is carried out over all scales and image locations. The locations of keypoints are determined in

the second stage. A measure of stability is used to select the keypoints. Based on the local image gradient directions, each keypoint is provided with an orientation in the third stage. In the fourth or last stage, a descriptor for each keypoint is constructed. Each keypoint contains the (x, y) coordinate of the features in the image, scale σ, orientation ρ and also the keypoint descriptor vector. Thus, a SIFT keypoint $\mathbf{f}$ can be denoted as follows,

$$\mathbf{f} = \{f_x, f_y, f_\sigma, f_\rho, f_{kpd}\} \tag{4.2}$$

The homing algorithm developed by Churchill and Vardy [12] is briefly described below:

The current view image and the goal image are given as input to the visual homing algorithm; the home direction is then computed from these two images. The SIFT algorithm is able to extract features from an image which are invariant to image rotation and scale. The matched features are found between the current view image and the goal image using the SIFT algorithm. Based on these SIFT features, two regions can be found in an image: a region of expansion and a region of contraction. According to this homing algorithm, the home direction will be aligned with the center of the region of contraction in the current view image. The above algorithm can be described in detail using figure 4.5. In this figure, CV refers to the image taken by the robot at its current position and SS refers to the image taken at the goal position. If the robot moves from position SS to position CV, the distance from the robot to feature A will increase; actually the distance from the robot to any feature on the same side of the perpendicular bisector of the line joining SS and CV will increase. On the

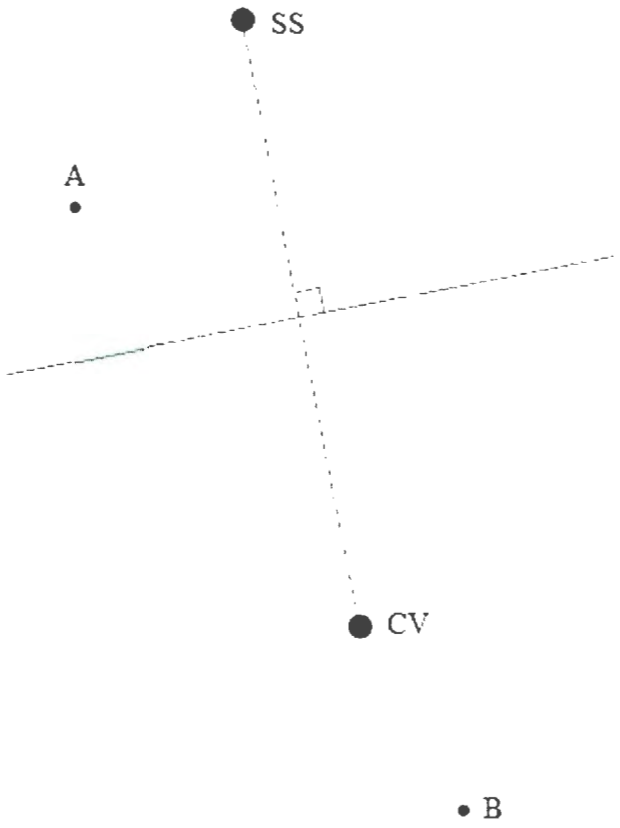

Figure 4.5: Robot pose diagram. Courtesy: Churchill and Vardy [12].

other hand, the distance from the robot to feature B will decrease; and this condition is true for all features on the same side of feature B. The assumption of the homing algorithm is that this change in distance will be reflected in a corresponding change in the scale parameter of the SIFT feature vector; the scale factor between two adjacent images differs by the multiplicative of a constant. Clearly the image features can be divided into two groups - expanding and contracting. If there are enough features distributed evenly on either side of the perpendicular bisector of the line joining SS and CV, then approximately half of them should experience expansion; while the other

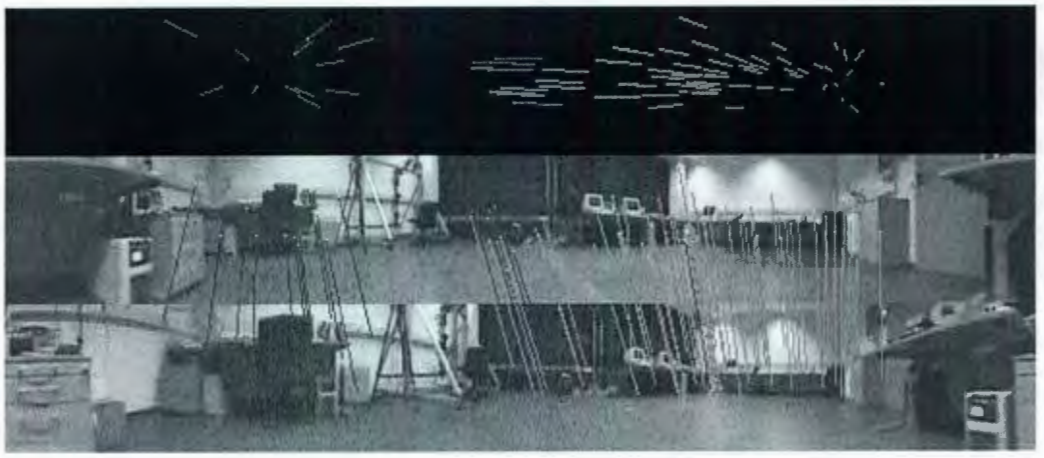

Figure 4.6: SIFT matched correspondences between CV (above) and SS (below), red lines refer to the features that have contracted from SS to CV and green lines refer to the features that have expanded from SS to CV. Above, regions of the image are shown more intuitively when vectors are mapped onto a single image. Courtesy: Dave Churchill and Andrew Vardy [12].

half should experience contraction; as a result there will be two regions - a region of contraction and a region of expansion. According to [12], the home direction will be aligned with the center of the region of contraction (on the assumption of the features being distributed uniformly throughout the environment).

In order to obtain the home direction, the center of the region of contraction or expansion from SS to CV with respect to CV must be calculated first. According to [12], in order to find the center of the region of contraction or expansion, the change in the feature size from SS to CV must be detected first i.e. whether a feature has shrunk or become larger from SS to CV. It can be observed from equation 4.2 that along with other information, each SIFT feature vector gives the information of scale σ at which it was detected; the magnitude of the scale parameter is directly related

to the size of the feature in the image.

Figure 4.6 shows the SIFT matched correspondences between CV (above) and SS (below). In the first part of figure 4.6, red lines refer to the features that have contracted from SS to CV and green lines refer to the features that have expanded from SS to CV, regions of the image are shown more intuitively when vectors are mapped onto a single image. It can be observed from the figure that the matches on the left side show that the size of the features has decreased from SS to CV (the chair seems to be larger in SS and much smaller in CV). Conversely, the matches on the right side show that the size of the features has increased from SS to CV (the computers look smaller in SS and much larger in CV). Since the area near the chair refers to the region of contraction in CV, this is the home direction the robot should move to get to the goal location.

The home direction θ_{homing} can be obtained as follows,

$$\theta_{homing} = atan2(\overline{s}, \overline{c}) \tag{4.3}$$

where,

$$\overline{s} = |M_{pos}| \sin(\theta_{pos}) + |M_{neg}|(\sin(\theta_{neg}) + \pi) \tag{4.4}$$

$$\overline{c} = |M_{pos}| \cos(\theta_{pos}) + |M_{neg}|(\cos(\theta_{neg}) + \pi) \tag{4.5}$$

In the above equations, M_{pos} and M_{neg} denote the SIFT matched correspondence features which are divided into these two groups based on the contracting and expanding features respectively; θ_{pos} and θ_{neg} denote the center of the region M_{pos} and

M_{neg} respectively. The value of θ_{homing} denotes the home direction with respect to the robot reference frame; the robot should be moved to this home direction to get to the goal location.

4.4 Rotation Estimation

In our exploration system, when the robot homes back to a previously created node, it continues to explore from this node using the remaining exploration directions of this node. But the exploration directions are calculated based on the heading of the robot when the node was first created, not based on the current heading of the robot. So in order to use the remaining exploration directions properly, we must find out the angular difference between the current heading and the previous heading of the node. We have used the method developed by Zeil *et al.* [68] to estimate the angle difference.

Zeil *et al.* [68] introduced a simple homing algorithm based on gradient descent. The method was derived from the observation that the difference between the current view and the snapshot image gradually increases with spatial distance; further, they demonstrated that a compass estimate can be obtained from a simple correlation of images. In their method, the home direction was determined from the gradient of the root mean square (RMS) difference between the current image and the snapshot; the orientation or rotation information was derived from the minimum of a rotational RMS function.

In order to calculate the root mean square (RMS) difference value, first the pixel-by-pixel differences between the two images are squared. Then the root of the mean

squared difference is obtained which gives the the RMS value for that image pair. One image is specified as the reference or snapshot image, then a number of images are taken from different positions surrounding the snapshot image position and the RMS difference value is calculated for each case. By plotting the RMS difference values, Zeil *et al.* [68] showed that the image differences change smoothly with physical distance from a reference position. Further they showed that if a snapshot image and the current view image have the same orientation, then the RMS pixel difference will be minimum.

In our system, rectangular panoramic images (fig.5.2) are used to compute the rotation angle. The value of a pixel at column i and row j of an image I is indicated by $I(i,j)$. The width and height of the image are denoted by w and h respectively. If two planar images are taken at the same position with different orientations, then the difference in their orientation can be expressed by a horizontal shift. Let I_θ denote the image captured at a certain position with orientation θ, then $I_\theta(i,j) = I_{\theta+\Delta\theta}(i+k,j)$. The rotation angle $\Delta\theta$ corresponds to a shift of k pixels. The relationship between $\Delta\theta$ and k is given below:

$$\Delta\theta = -k\frac{2\pi}{w} \tag{4.6}$$

We have used the notations in this section from the paper [63] which is also on visual compass. The rotation angles are measured counter-clockwise from the robot's forward heading, but image indices increase from left to right; for this reason the negative sign is used in equation 4.6.

Let S denote the image captured at the goal position and C denote the current

view image. Then the root mean square (RMS) difference between the current view image and the snapshot image, for a rotation by k pixels, can be obtained as:

$$SD(i,j,k) = [C(i+k,j) - S(i,j)]^2 \tag{4.7}$$

The sum of all values in SD is used to calculate the image distance function for shift k ,

$$ssd(k) = \sum_i \sum_j SD(i,j,k) \tag{4.8}$$

According to [68], the value of horizontal shift $k^{'}$ that minimizes ssd is considered as the angle difference (in pixels) between the current view image and the snapshot image,

$$k^{'} = \arg \min_{k \in [0,w-1]} ssd(k) \tag{4.9}$$

In this way, the angle difference between the current heading and the previous heading of any node is obtained; the robot turns according to this angle and then continues exploring the rest of the environment using the remaining exploration directions of the node.

4.5 Path Planning

While exploring the environment, there may occur situations where the robot homes to a node with no more exploration directions i.e. all its exploration directions are already explored by the robot; but other nodes with unexplored directions still exist

in the map. Path planning is used in such a situation so that the robot can find the nearest node with unexplored direction and then plans a path from the current node to that node. There can be a number of paths from one node to anther one, one path may be longer than the other one; so it is important to find out the shortest path to make the system efficient. In our method the shortest path is determined by using Dijkstra's shortest path algorithm [16], which is a well known algorithm for finding the shortest path between two nodes. According to this algorithm, a path from a source vertex v to a target vertex u is said to be the shortest path if its total cost is minimum among all v-to-u paths. In this algorithm, the shortest path is generated based on the edges of the nodes; in other words, the already visited routes of the topological map. Weights of all edges are equal in our system.

Our path planning strategy is described as follows: first, paths from the current node to all the other nodes are calculated using Dijkstra's shortest path algorithm. Then the nodes are sorted in an array starting from the node with the shortest path from the current node. Next, the first node in the array is checked to see if any unexplored direction exists in it. If the node contains unexplored direction, then it is chosen as the node to be homed from the current node. On the other hand, if all the directions are explored for the first node in the array, then the next nodes are gradually checked and the first node with unexplored direction is chosen as the node to be homed from the current node. The path from the current node to this nearest node (with unexplored direction) is already planned using Dijkstra's shortest path algorithm. Finally, visual homing is used to home to each node along the path in sequence.

4.6 Navigating between places

In this section we describe our strategy to navigate from any node to a goal node in the topological map. No geometric information is used for the navigation purpose. In our system, topological navigation is performed using visual information only. So the robot has no idea about the distance from the starting node to the goal node. One method to find the distance is by finding corresponding features in three or more images; this is a very common technique in the field of visual servoing [42]. But the method is not efficient in case of dynamic environments as it becomes very difficult to find stable correspondences among three images. Instead we used Dijkstra's shortest path algorithm [16] to calculate the distance or path between the starting node and the goal node. All edges in the graph are assumed equal in their distances. In the navigation state, the user first inputs the goal node; our strategy is that any node in the map can be chosen as the goal node. Then the starting node and goal node are given as the input to Dijkstra's algorithm. The algorithm calculates all the paths from the starting node to the goal node (if there exists more than one path from the starting node to the goal node); then the shortest path is given as the output of the algorithm.

Our navigation strategy directs the mobile agent towards one node at a time i.e. the robot performs homing to reach the next node in the path given by Dijkstra's algorithm, this intermediate node can be referred to as a subgoal on the path to the goal node. After reaching the current subgoal, the robot tries to home to the next subgoal from the current one; in this way, it reaches the goal node which indicates the completion of the navigation process. The homing method used to reach the subgoal

as well as the goal node is described in detail in section 4.3.

4.7 Robot Control and Obstacle Avoidance

We used the built-in obstacle avoidance system of the pioneer robot which uses the laser range finder to detect obstacles. If an obstacle is detected at less than 20 cm distance, the robot would turn 15^0 away from the obstacle's bearing as decided by the obstacle avoidance system; in this way, the mobile agent is able to avoid obstacles in the environment. The pioneer robot was controlled using a software named ARIA - an object-oriented, robot control applications-programming interface for intelligent mobile robots. ARIA provides the higher-level action system. Actions are individual objects that independently provide motion requests which are evaluated and then combined each cycle to produce a final set of movement commands. Actions are evaluated by the robot's action resolver in descending order of priority (highest priority first, lowest priority last). Actions can be created according to the user's requirements. In our system, we created the exploration action and the navigation action in order to achieve the research objectives properly. The obstacle avoidance action was given the highest priority; the exploration action and the navigation action were given low priority.

Chapter 5

Experimental Results

5.1 Image processing

All experiments were done using a Pioneer 3AT mobile robot equipped with a SICK laser range finder and an omnidirectional camera system. The camera was a catadioptric system consisting of an upward looking camera with a hyperbolic mirror mounted above it. The hyperbolic mirror expanded the camera's field of view to allow the capture of omnidirectional images. All explorations were autonomous i.e. there was no manual control of the robot. The exploration and navigation experiments were done in an office environment, in the Intelligent Systems Lab within the Faculty of Engineering and Applied Science at Memorial University of Newfoundland.

In our system, omnidirectional colored images of the environment were captured and converted to gray level image as can be seen from figure 5.1. Then the gray level omnidirectional images were hyperbolically mapped to produce panoramic images. An image unfolding procedure was applied which finds the projection of the original

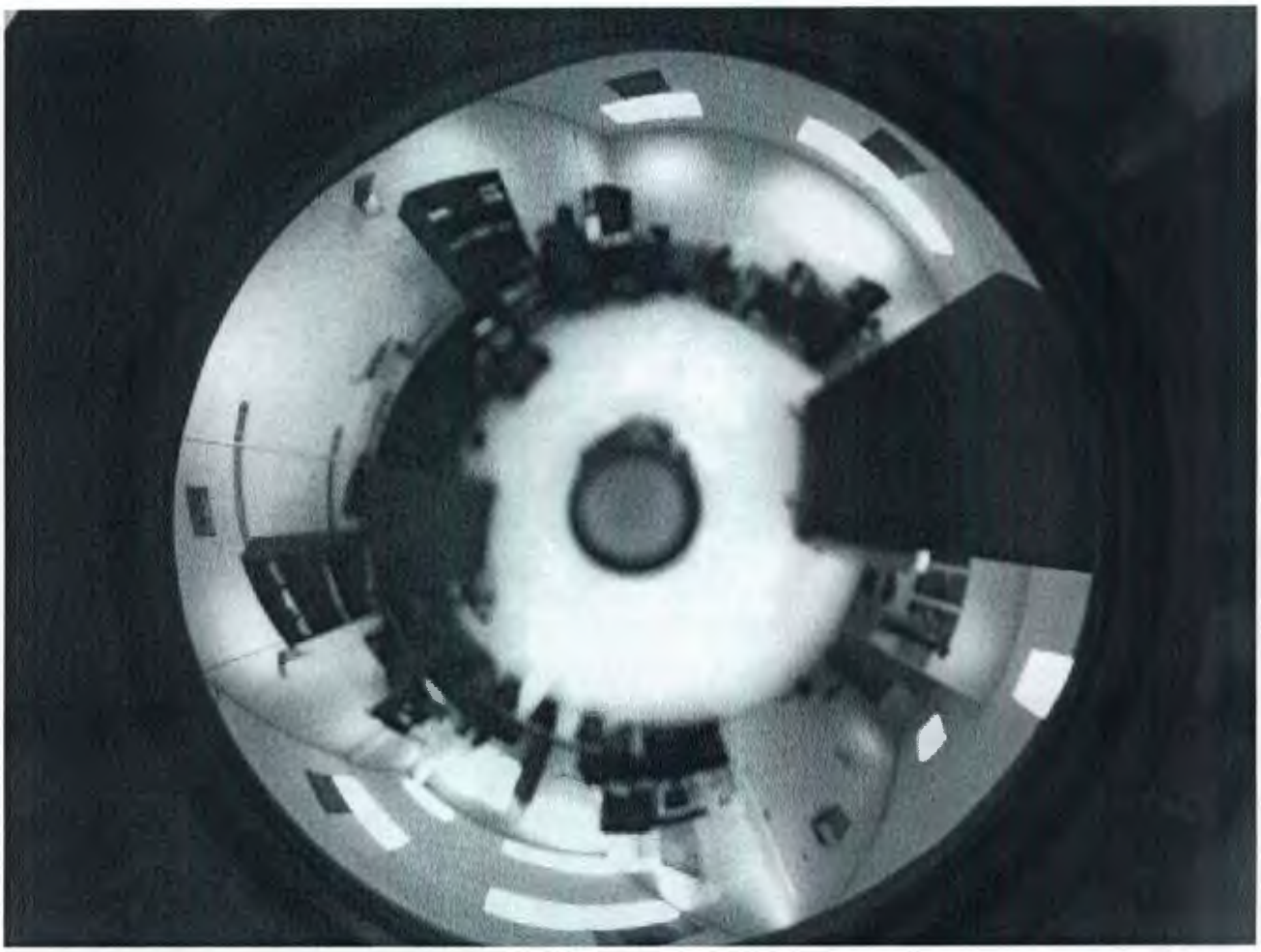

Figure 5.1: Sample omnidirectional gray scale image.

image onto a sphere centered at the upper focus of the hyperbola. Pixels from the original gray level omnidirectional image are mapped to a rectangular output image which has rows and columns corresponding to the elevation (vertical angle) and azimuth (horizontal angle) of the spherical projection. The unfolded panoramic image is shown in figure 5.2. The gray level panoramic image was stored at each node.

Figure 5.2: unfolded panoramic image obtained by hyperbolically mapping figure 5.1.

5.2 Exploration Results

5.2.1 Parameters Varied

Different topological maps of the environment were obtained with various settings of two key parameters, namely - *forward_distance* and *hist_threshold*. In our exploration system, after turning according to the exploration direction, the next task of the mobile robot is to move forward (see section 4.1 for details); the amount of the forward distance moved by the robot is controlled by the parameter *forward_distance*. The second parameter *hist_threshold* is used in the image matching state of our exploration system. We have used the dissimilarity measure to compare the images as described in chapter 3; a dissimilarity value of zero means that the two images are perfectly similar; as the value of the dissimilarity measure increases, the images are becoming more dissimilar from each other. The parameter *hist_threshold* refers to the threshold dissimilarity value for which the images will be considered as similar enough i.e. their locations in the environment are close to each other. The topological map of figure 5.3 was obtained with a *forward_distance* of 2 m and *hist_threshold* of 600. So for figure 5.3, if the dissimilarity value is smaller than or equal to 600, then the two images will be considered visually similar.

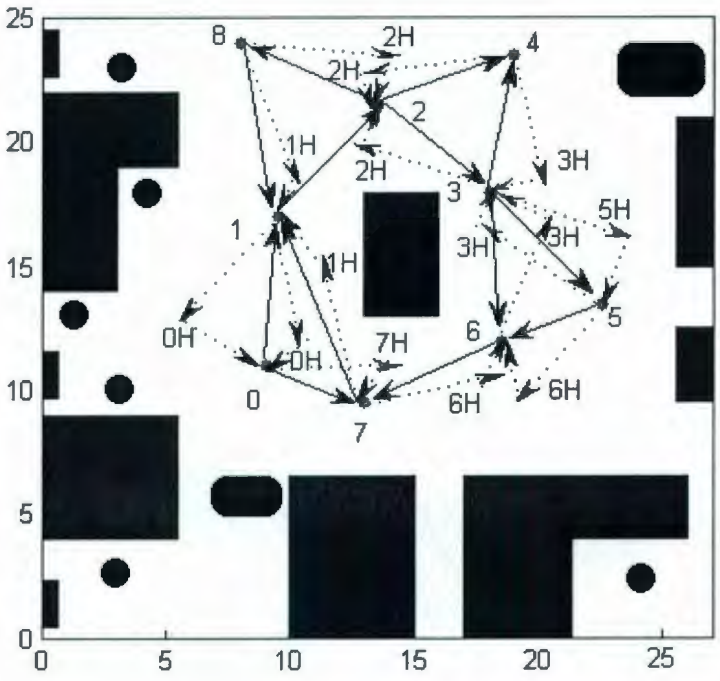

Figure 5.3: Topological map of the environment, *forward_distance* 2 m and *hist_threshold* 600

5.2.2 Topological Maps

The purpose of our experiments is to verify the performance of the implemented exploration and navigation system. In order to test our system properly, we performed the experiments with a mobile agent in an office environment. The topological maps were built with different sets of values of the parameters *forward_distance* and *hist_threshold*. As stated earlier, the topological map of figure 5.3 was obtained with a *forward_distance* of 2 m and *hist_threshold* of 600. This topological map contains 9 nodes and 13 edges. In this figure, the black squares refer to all the tables, book shelves and file cabinets in the office; the circles refer to chairs in the office. The red dots refer to the nodes of the topological map; solid lines refer to the edges between

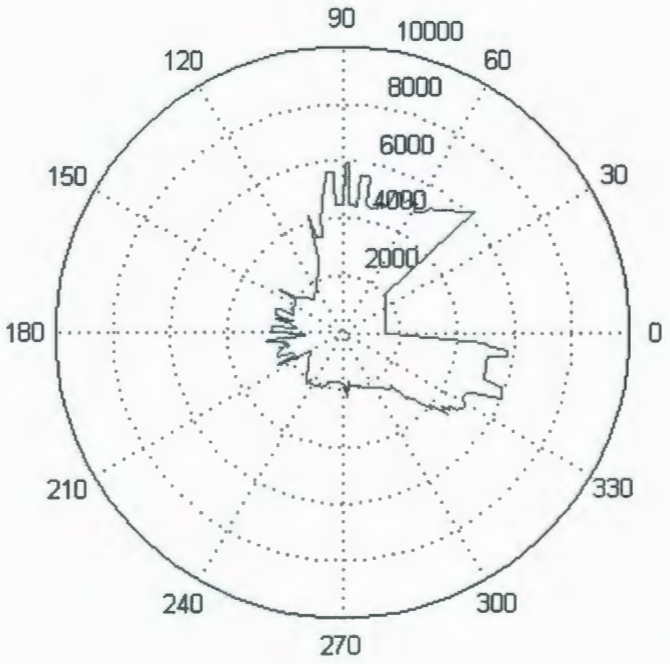

Figure 5.4: Laser data for node 0 of the topological map of figure 5.3. Exploration directions are 83 and -22.5(in degrees).

the nodes and the dotted lines refer to the homing attempts between the nodes. All the nodes in the map are represented with numbers and the numbers with a H refer to the places where the homing state was activated. Different states, nodes and dissimilarity values of this mapping run are shown in detail in table 5.1. The full description of the exploration run for figure 5.3 is given below:

The robot was placed at the lower left corner of the environment as can be seen from figure 5.3, the exploration started from that location; so at the beginning there was no topological map stored in the memory of the robot. First the robot created the first node (node 0) of the topological map; at each node, the exploration directions and a snapshot at the current location were stored. So the 360 degree laser data from

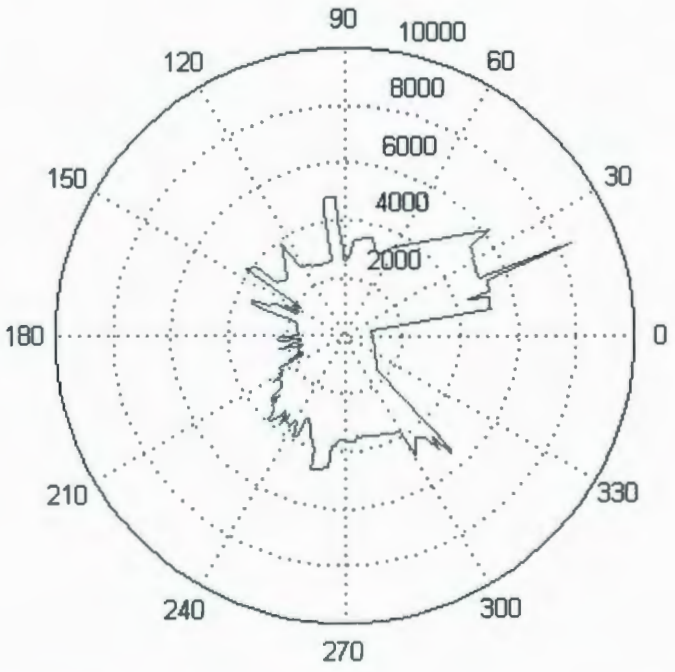

Figure 5.5: Laser data for node 1 of the topological map of figure 5.3. Exploration directions are 32.5, -58.5 and -102.5(in degrees).

the current location was taken and an exploration direction list was created from the laser data; figure 5.4 shows the laser data for node 0, exploration directions were calculated according to the exploration strategy described in section 4.1. The best four exploration directions are stored at each node (less if there are fewer than four exploration directions from that node). Two exploration directions were calculated for node 0 by our exploration algorithm, they were 83 and -22.5 (in degrees). The robot turned according to the first exploration direction stored at node 0; then it moved forward 2 m, after that the robot checked if the current view was similar to any known node; the output was negative which means the current view was not similar to any known node, so a new node (node 1) was created.

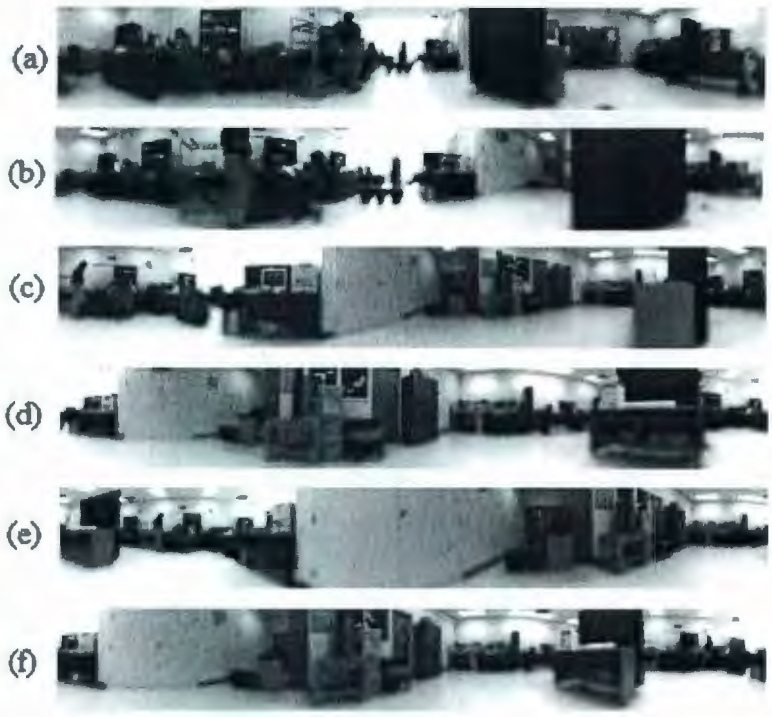

Figure 5.6: (a) Node image 0, (b) Node image 1, (c) Node image 2, (d) Node image 3, (e) Node image 4, (f) Current view after moving forward 2m from node 4 of the topological map of figure 5.3. Value of dissimilarity measure is 1354.18, 1396.63, 707.94, 218.215 and 949.953 with node image 0, 1, 2, 3 and 4 respectively.

As before, a snapshot was taken at the current location and 360 degree laser data was taken for this location, then exploration directions were obtained from the laser data. Figure 5.5 shows the laser data for node 1 of the topological map of figure 5.3; exploration directions were 32.5, -58.5 and -102.5 (in degrees). Our exploration algorithm automatically creates an edge between the current node and the previous node; the edges of our map are bidirectional i.e. if there is an edge between nodes 0 and 1, then the robot can traverse the path from node 0 to 1 and similarly from

node 1 to 0. So an edge was created between node 0 and 1; then the robot turned according to the first exploration direction stored at node 1 and moved forward 2 m; the image similarity was checked and the output was negative, so node 2 was created. Exploration directions for node 2 were 7 and -106 (in degrees); an edge was created between nodes 1 and 2. Similarly, node 3 was created with exploration directions -102, 73, 14.5, 111 and -71.5 (in degrees) and an edge was created between nodes 2 and 3. After that the robot continued exploring the environment from node 3 by turning according to the first exploration direction stored at this node. After moving forward 2m, the output of the image similarity method was negative; so node 4 was created, exploration directions were -63 and -145, and an edge was created between nodes 3 and 4.

When the image similarity was checked after the robot moved forward 2 m from node 4, the output was positive and the the current view was similar to node 3, the dissimilarity value was 218.215. Figure 5.6 shows all the five node images of the topological map (currently the map consists of these five nodes) and also the current view. It can be observed from figure 5.6(d) and figure 5.6(f) that the current view is quite similar to the node image 3; which justifies our image similarity algorithm. According to our algorithm, the robot would not create a node at this location, instead it would home back to the similar node from the current location; so the homing state began and the robot homed back to node 3 from node 4; this means that the path between node 4 and 3 is traversable. Now the robot should continue exploration from the current node 3 with the next best exploration direction stored at this node, but the exploration directions stored at node 3 are based on the previous heading of this node and the current heading could be different from the previous heading; so the

State	Node	Dissimilarity value
Exploration	0 – 1	–
Exploration	1 – 2	–
Exploration	2 – 3	–
Exploration	3 – 4	–
Exploration(Homing)	4 – 3	218.215
Exploration	3 – 5	–
Exploration(Homing)	5 – 3	141.214
Exploration	3 – 6	–
Exploration(Homing)	6 – 3	144.173
Exploration(Homing)	3 – 5	193.405
Exploration(Homing)	5 – 6	400.385
Exploration	6 – 7	–
Exploration(Homing)	7 – 1	292.413
Exploration(Homing)	1 – 0	501.859
Exploration(Homing)	0 – 7	380.358
Exploration(Homing)	7 – 6	220.192
Path planning	6 – 3	–
Exploration(Homing)	3 – 2	250.419
Exploration	2 – 8	–
Exploration(Homing)	8 – 1	296.851
Exploration(Homing)	1 – 0	193.297
Path planning	0 – 1 – 8	–
Exploration(Homing)	8 – 2	344.907
Path planning	2 – 4	–
Exploration(Homing)	4 – 2	222.561
EXPLORATION DONE	–	–

Table 5.1: Different states, nodes and dissimilarity values of the mapping run with *forward_distance* 2m and *hist_threshold* 600.

angle difference between the current heading and the previous heading of this node was calculated using the current view image and the node image 3; and the robot turned to this angle. Then the robot continued exploration from node 3 with the next best exploration direction stored at this node.

After moving forward from node 3, the image similarity was checked and the output was negative, so node 5 was created with exploration directions -66.5 and 8.5; an edge was created between node 3 and 5. When it moved forward from node 5, the current view was found similar to node 3, the dissimilarity value was 141.214; so the homing state began and the robot homed back to node 3; the angle difference between the current heading and the previous heading of this node was calculated as before and the robot was turned according to this angle. Then it continued exploration from node 3 with the next best exploration direction stored at this node. Then node 6 was created with exploration directions -123 and 9, an edge was created between node 6 and 3. After moving forward from node 6, the current view was found similar to node 3, the dissimilarity value was 144.173; so the robot homed back to node 3 and the angle difference was calculated and the robot was turned according to the angle. After moving forward from node 3, homing state began as the current view was found similar to node 5; then after performing remaining states and moving forward, the current view was found similar to node 6, so homing state began once again; remaining states followed. When the image similarity was checked after the robot moved forward 2 m from node 6, the output was negative.

So node 7 was created with exploration directions -11.5 and -80; an edge was created between node 7 and 6. The current view was found similar to node 1 after moving froward from node 7, the dissimilarity value was 292.413; so it homed back

to node 1. Three homing states followed after that: the robot homed from node 1 to node 0, then from node 0 to node 7 and from node 7 to node 6 afterwards; the dissimilarity values are given in table 5.1. Now the robot should continue exploration from the current node 6 with the next best exploration direction stored at this node, but there was no more exploration direction remaining at node 6, so the robot checked if there was any exploration direction remaining at any node of the map and the output was positive i.e. exploration direction still existed at other node of the map; so now the robot should find the node whose exploration direction still existed and then home to that node. Path planning state began to perform this task. By using Dijkstra's shortest path algorithm, the nearest node (with exploration direction) to node 6 was found to be node 3 and the path was $6 - 3$; the robot then went to node 3 by homing (The homing attempts for path planing are not shown in figure 5.3 to keep it less obstructed with lines). Then it explored from node 3 with the last exploration direction stored at node 3; the current view was found similar to node 2 with dissimilarity value of 250.419. It then explored from node 2 and node 8 was created with exploration directions 178 and -56.5. After moving forward from node 8, the current view was found similar to node 1, the dissimilarity value was 296.851; so it homed to node 1. Homing state began again after moving forward from node 1 as the current view was found similar to node 0 with dissimilarity value of 193.297.

As there was no more exploration direction remaining at node 0, the robot checked if there was any exploration direction remaining at any node of the map and the output was positive; so path planning began again. The nearest node (with exploration direction) to node 0 was found to be node 8 and the path executed was $0 - 1 - 8$. Then it explored from node 8 with the last exploration direction stored at node 8, the

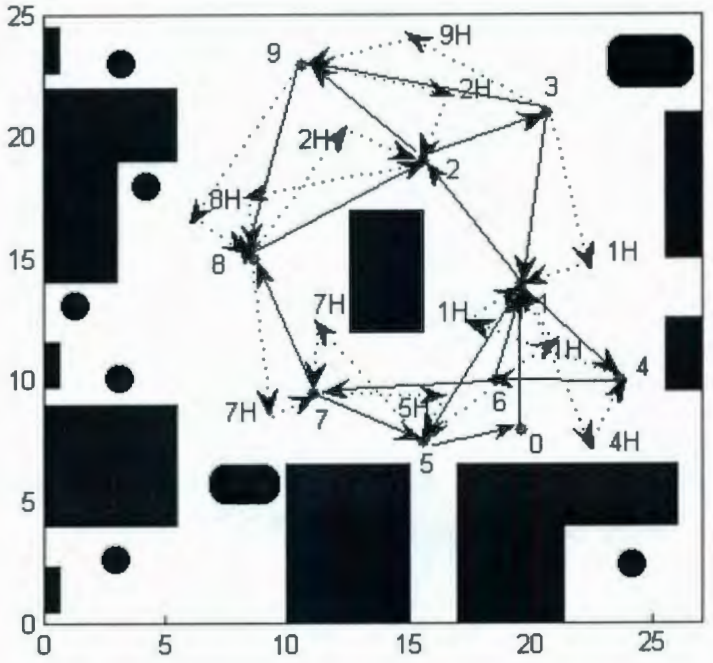

Figure 5.7: Topological map of the environment, *forward_distance* 2m and *hist_threshold* 500

current view was found similar to node 2. As there was no more exploration direction remaining at node 2, path planning state began and the nearest node was found to be node 4 and the path was $2-4$, the robot homed to node 4 and continued exploration from node 4; then the current view was found similar to node 2 with dissimilarity value of 222.561; it homed back to node 2. As there was no more exploration direction remaining at node 2, the robot checked if there was any exploration direction remaining at any node of the map and the output was negative indicating the end of exploration. In this way, the exploration was done using a laser range finder and an omnidirectional camera; and a topological map of the environment was created.

The topological map of figure 5.7 was obtained with the *forward_distance* of 2m

State	Node	Dissimilarity value
Exploration	0 – 1	–
Exploration	1 – 2	–
Exploration	2 – 3	–
Exploration(Homing)	3 – 1	271.613
Exploration	1 – 4	–
Exploration(Homing)	4 – 1	223.302
Exploration	1 – 5	–
Exploration(Homing)	5 – 1	210.528
Exploration(Homing)	1 – 4	462.229
Exploration	4 – 6	–
Exploration	6 – 7	–
Exploration(Homing)	7 – 5	201.305
Exploration(Homing)	5 – 7	134.912
Exploration	7 – 8	–
Exploration(Homing)	8 – 2	256.461
Exploration	2 – 9	–
Exploration(Homing)	9 – 2	411.596
Exploration(Homing)	2 – 8	169.943
Path planning	8 – 7 – 6	–
Exploration(Homing)	6 – 1	355.416
Path planning	1 – 3	–
Exploration(Homing)	3 – 9	409.22
Exploration(Homing)	9 – 8	274.245
Exploration(Homing)	8 – 7	251.64
Path planning	7 – 5 – 0	–
Exploration(Homing)	0 – 5	327.59
EXPLORATION DONE	–	–

Table 5.2: Different states, nodes and dissimilarity values of the mapping run with *forward_distance* 2m and *hist_threshold* 500.

State	Node	Dissimilarity value
Exploration	$0-1$	–
Exploration	$1-2$	–
Exploration	$2-3$	–
Exploration(Homing)	$3-2$	144.945
Exploration	$2-4$	–
Exploration(Homing)	$4-2$	205.025
Path planning	$2-1$	–
Exploration(Homing)	$1-0$	302.578
Exploration	$0-5$	–
Exploration(Homing)	$5-1$	289.07
Path planning	$1-0$	–
Exploration	$0-6$	–
Exploration(Homing)	$6-0$	299.044
Path planning	$0-5$	–
Exploration	$5-7$	–
Exploration(Homing)	$7-6$	291.696
Exploration	$6-8$	–
Exploration	$8-9$	–
Exploration(Homing)	$9-4$	322.546
Exploration(Homing)	$4-9$	345.34
Exploration(Homing)	$9-8$	187.398
Exploration(Homing)	$8-7$	255.66
Exploration(Homing)	$7-8$	290.314
Path planning	$8-9-4$	–
EXPLORATION DONE	–	–

Table 5.3: Different states, nodes and dissimilarity values of the mapping run with *forward_distance* 1.5m and *hist_threshold* 500.

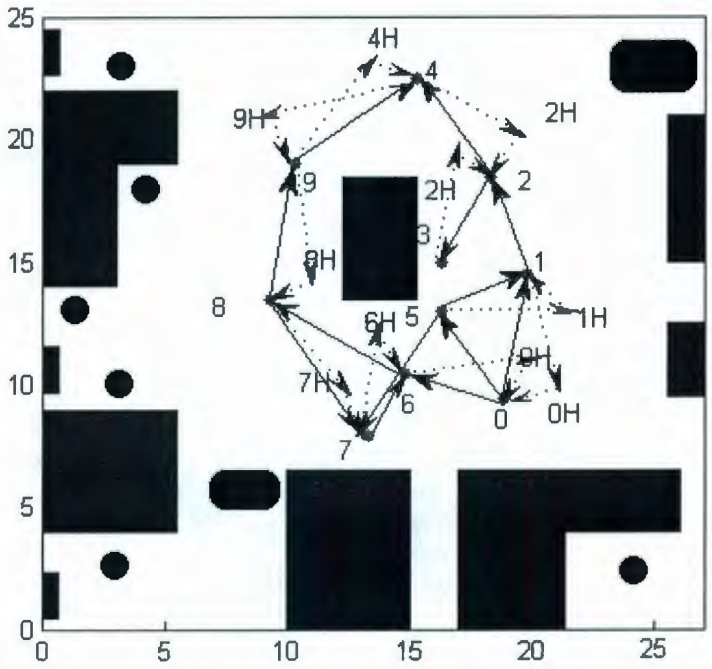

Figure 5.8: Topological map of the environment, *forward_distance* 1.5m and *hist_threshold* 500.

and the *hist_threshold* in the image comparison method was 500; the exploration run was successful and a complete topological map of the environment was created. The main difference of this map with the previous map of figure 5.3 is that the threshold of dissimilarity measure is less for the current map; as a result the exploration algorithm will create a new node if the value of the dissimilarity measure is greater than 500; so nodes will be closer to each other as compared to the previous map. Different states, nodes and dissimilarity values of this map are shown in detail in table 5.2. This map contains 10 nodes and 15 edges. The starting position of this map was in the lower right corner of the room which is different from that of the map of figure 5.3; as a result, the exploration paths are also different.

State	Node	Dissimilarity value
Exploration	0 – 1	–
Exploration	1 – 2	–
Exploration	2 – 3	–
Exploration	3 – 4	–
Exploration(Homing)	4 – 3	191.394
Exploration	3 – 5	–
Exploration(Homing)	5 – 3	234.894
Exploration	3 – 6	–
Exploration(Homing)	6 – 4	291.058
Exploration	4 – 7	–
Exploration(Homing)	7 – 6	184.638
Exploration	6 – 8	–
Exploration(Homing)	8 – 4	341.646
Path planning	4 – 7	–
Exploration	7 – 9	–
Exploration(Homing)	9 – 0	359.473
Exploration(Homing)	0 – 9	335.91
Path planning	9 – 7 – 8	–
Exploration	8 – 6	–
Path planning	6 – 3 – 5	–
Exploration(Homing)	5 – 2	402.11
Exploration(Homing)	2 – 1	276.45
Exploration(Homing)	1 – 0	356.432
EXPLORATION DONE	–	–

Table 5.4: Different states, nodes and dissimilarity values of the mapping run with *forward_distance* 1.5m and *hist_threshold* 400.

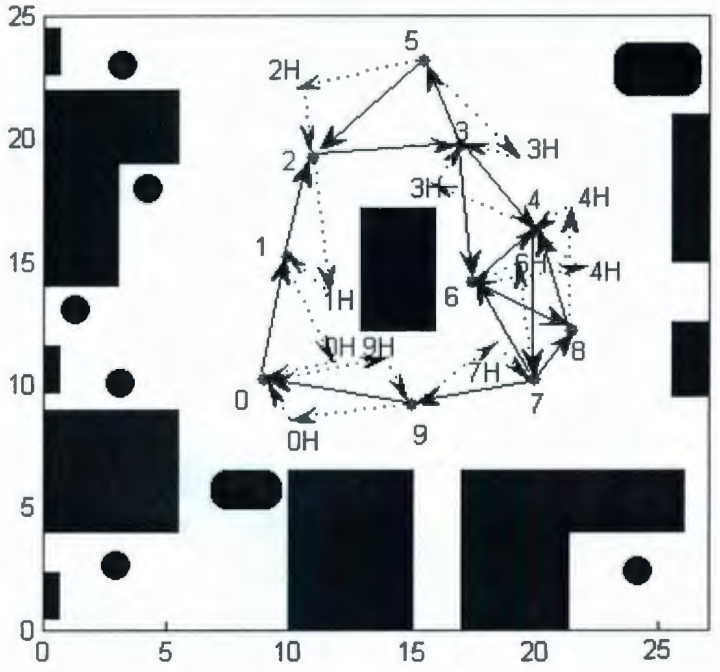

Figure 5.9: Topological map of the environment, *forward_distance* 1.5m and *hist_threshold* 400.

The topological map of figure 5.8 was obtained with the *forward_distance* of 1.5m and the *hist_threshold* was 500; the exploration run was completed properly and a topological map of the environment was created. The *forward_distance* for this map is lower than the previous two maps, so the condition whether the current view is similar to known node is checked more frequently in this map than the previous maps. *hist_threshold* values more than 500 were not used for the *forward_distance* of 1.5m, because if the value of *hist_threshold* was increased to 600 or higher, then most of the times the current view was found similar to the previous node after moving forward 1.5m; The states, nodes and dissimilarity values of this run are shown in detail in table 5.3. This map contains 10 nodes and 14 edges. The starting position for this

run was the lower right corner of the room.

The topological map of figure 5.9 was obtained with the *forward_distance* of 1.5m and the *hist_threshold* in the image comparison method was 400; the exploration run completed properly and a topological map of the environment was created. The *hist_threshold* for this map is lower than the previous maps. The states, nodes and dissimilarity values of this run are shown in detail in table 5.4. This map contains 10 nodes and 14 edges. The starting position of this map was the lower left corner of the room. It can be observed from figure 5.9 that nodes 7 and 8 were created very close to each other; this happened because the *hist_threshold* is only 400 for this map, so if the dissimilarity value is even slightly above 400, another node will be created. For this reason node 8 was created very close to node 7 instead of homing back to node 7.

5.2.3 Limitations

Like other methods, our exploration system has some limitations too. One limitation is that the image comparison method which is based on histograms, may denote that the current view is similar to a node image, but in reality the current view may be similar to some other node image. Such a situation occurred in the case of the topological map of figure 5.10. The topological map of figure 5.10 was obtained with the *forward_distance* of 2m and the *hist_threshold* 400. The states, nodes and dissimilarity values of this run are shown in table 5.6.

It can be observed from figure 5.10 that during the exploration, after creating node 8, when the robot moved forward and image comparison was done to check

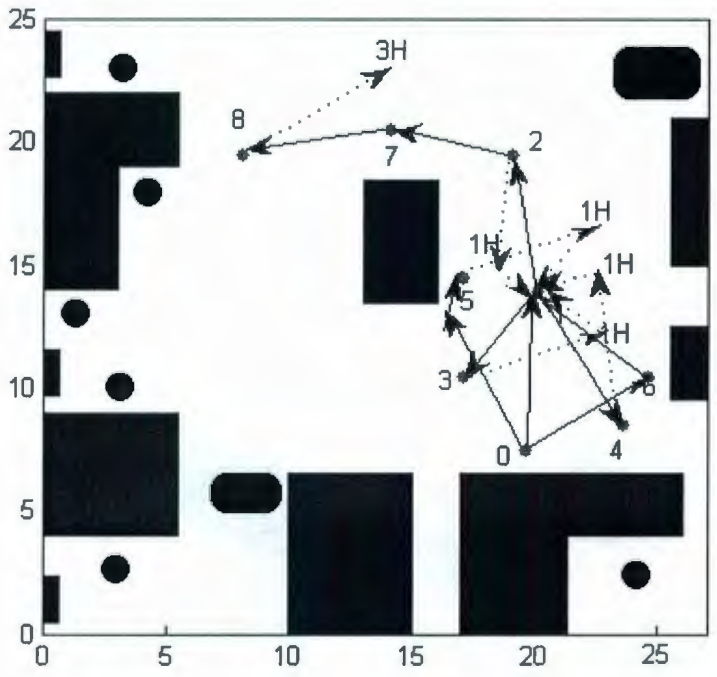

Figure 5.10: Topological map of the environment, *forward_distance* 2m and *hist_threshold* 400

the similarity of the current view with any of the created nodes, the output of the image comparison method was that the current view was similar to node image 3; but actually the robot was much closer to node 7 than to node 3. The reason was that nodes 3 and 7 were created at two opposite side of an obstacle (the obstacle, placed in the middle of the room, was a study table) as can be seen from figure 5.10; as a result the images of node 3 and 7 had a lot of similarity; the images of node 3 and 7 are shown in figure 5.11. The dissimilarity measure for node image 3 was 397.597 and for node 7 it was 466.611 (see table 5.5); so the dissimilarity measures were also close. One solution to this problem is to use color histogram instead of gray level histograms; which may prevent some mismatches of this type. The Fourier transform

(a)

(b)

Figure 5.11: (a)node image 3 and (b) node image 7 of the topological mao with *forward_distance* 2m and *hist_threshold* 400

may be another solution for this problem. In the calculation of image dissimilarity of two omnidirectional images in the Fourier transform method, the magnitude of the rows of each image is taken; this is invariant to the rotation of the image around the optical axis. Thus, the Fourier transform method may be more robust against similar problems like the one stated above.

5.3 Navigation Results

When the exploration is complete and a topological map of the environment is built, then the robot can perform navigation to any node in the map. The user can input the goal node, any node in the map can be chosen as the goal node; then the robot will perform visual homing to reach the goal node. If the goal node is far from the current position of the robot, it performs homing to intermediate nodes along the path until it reaches the goal node.

Figure 5.12 shows a navigation run using the topological map with *forward_distance*

Node	Dissimilarity measure
0	944.223
1	695.344
2	713.68
3	397.597
4	1399.32
5	757.153
6	1786.98
7	466.611
8	869.829

Table 5.5: Dissimilarity measure - 2m 400

of 2m and the *hist_threshold* of 600 (the topological map is shown in figure 5.3). In figure 5.12, the goal node was given as node 2; the current node was node 6. First the system planned the path using the path planning algorithm, the shortest path between the current node and the goal node was given as $6 - 3 - 2$. So node 3 was the subgoal on the path to the goal node and the robot performed visual homing to home to the subgoal from the current node 6; after reaching the subgoal, it homed to the goal node (which was node 2) from the current node (which was node 3) using visual homing. The blue line indicates the navigation path in figure 5.12. Figure 5.13 shows a navigation run using the topological map with *forward_distance* of 1.5m and *hist_threshold* of 500 (the topological map is shown in figure 5.8). In figure 5.13, the goal node was given as node 8; the current node was node 4. The output of the path planning algorithm was $4 - 9 - 8$; the robot performed visual homing to reach the

State	Node	Dissimilarity value
Exploration	$0-1$	–
Exploration	$1-2$	–
Exploration(Homing)	$2-1$	390.121
Exploration	$1-3$	–
Exploration(Homing)	$3-1$	246.624
Exploration	$1-4$	–
Path planning	$4-1-0$	–
Exploration	$0-5$	–
Exploration(Homing)	$5-1$	228.322
Path planning	$1-0$	–
Exploration	$0-6$	–
Path planning	$6-1-2$	–
Exploration	$2-7$	–
Exploration	$7-8$	–
Exploration(Homing)	$8-3$	397.597

Table 5.6: Different states, nodes and dissimilarity values of the mapping run with *forward_distance* 2m and *hist_threshold* 400.

goal node as before.

State	Node
Navigation	$6-3$
Navigation	$3-2$
NAVIGATION DONE	–

Table 5.7: Different states and nodes of the navigation run for the topological map with *forward_distance* 2m and *hist_threshold* 600.

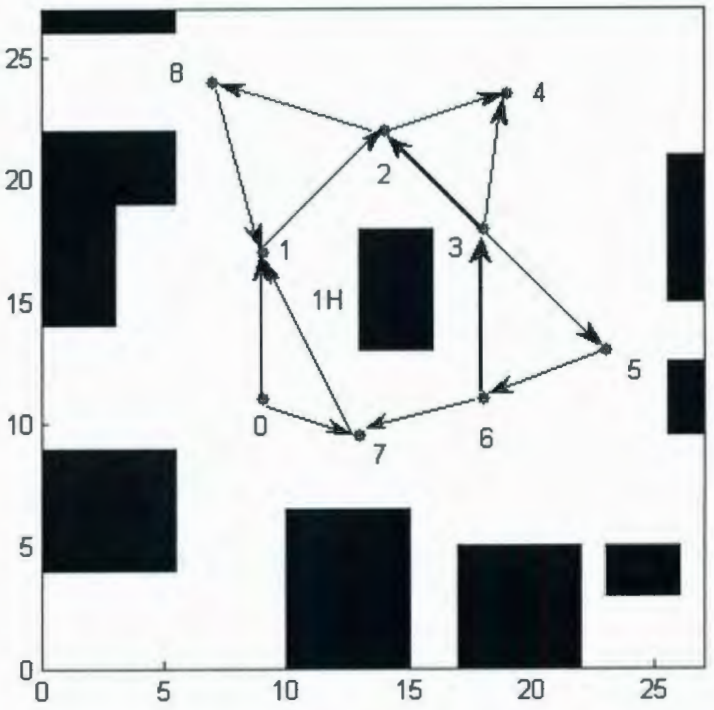

Figure 5.12: Navigation of the mobile robot using the topological map with *forward_distance* 2m and *hist_threshold* 600. The robot navigated from node 6 to node 2. Blue line shows the navigation path.

State	Node
Navigation	$4-9$
Navigation	$9-8$
NAVIGATION DONE	–

Table 5.8: Different states and nodes of the navigation run for the topological map with *forward_distance* 1.5m and *hist_threshold* 500.

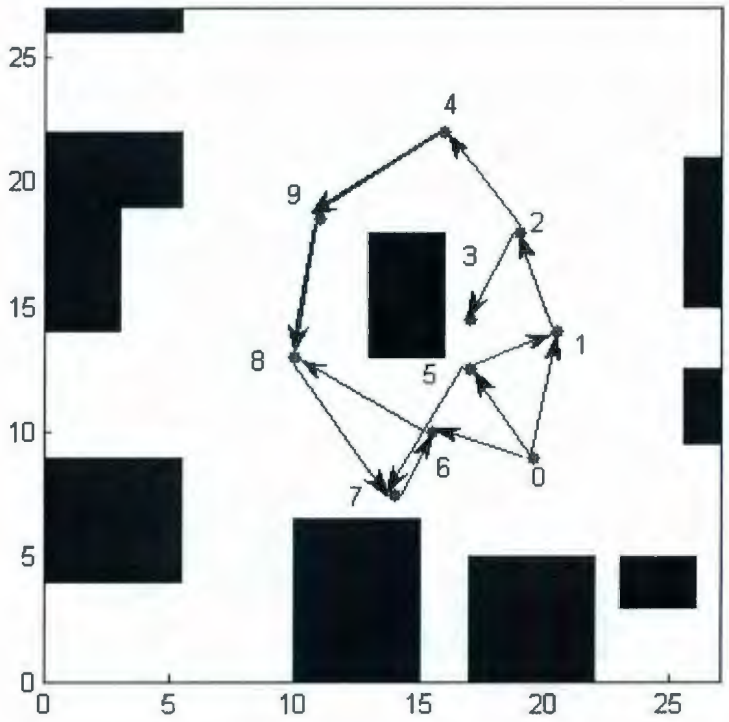

Figure 5.13: Navigation of the mobile robot using the topological map with *forward_distance* 1.5m and *hist_threshold* 500. The robot navigated from node 4 to node 8. Blue line shows the navigation path.

Chapter 6

Conclusion

We have presented a complete implementation of an autonomous exploration and navigation system in this thesis. An important contribution of this work was to develop a novel autonomous robot exploration and navigation system, using a topological representation of the environment. We have demonstrated that exploration and navigation can be done without storing precise metric information. A laser range finder and an omnidirectional camera were used to build the topological representation of the environment. We were able to build a topological map that satisfied the following requirements: (a) simple and easy to build, (b) can be built online in realtime, (c) does not utilize a large amount of memory and (d) uses only laser data and visual information. The time required for a mobile robot exploration depends on the size of the environment; in our autonomous exploration and navigation system, each iteration of the map building process required approximately 5-10 seconds.

A global image comparison technique was used to distinguish between different places. We compared the performance of four different global image comparison

techniques - three different histogram methods and the Fourier transform method. Two different experiments were done using a database of real images. It was observed that both the Fourier transform method and the Jeffrey divergence method, which is a histogram-based technique, performed well in the experiments. We chose to use the Jeffrey divergence method in the image matching section of the exploration and navigation system because of its simplicity and realtime computability.

We used only an omnidirectional camera for the navigation process. Once the map was built, the robot was able to navigate from its current node to any other node of the topological map. A visual homing mechanism was implemented in order to move the robot from one node to the next node. Our exploration and navigation system was implemented and tested on a Pioneer 3AT mobile robot; all the experiments were performed in an unmodified office environment. We were able to build an autonomous topological map online. Later when the map was completed, we tested our navigation method; the mobile robot was able to navigate to a goal node properly.

6.1 Future Work

One obvious future extension of this work is to test the whole system in outdoor environments; since all the biological navigations are happening in open air. The exploration strategy and the visual homing method may need some modifications in order to use them outdoor. Another extension can be the implementation of the navigation process in presence of illumination changes in the environment. Other future extensions include testing the system in more complex environments and solving the problem of perceptual aliasing in topological maps.

Bibliography

[1] N. Aihara, H. Iwasa, N. Yokoya, and H. Takemura. Memory-based self-localization using omnidirectional images. *Proceedings of the Fourteenth International Conference on Pattern Recognition*, 2:1799–1803, 1998.

[2] D. Amarasinghe, G. Mann, and R. Gosine. Integrated laser-camera sensor for the detection and localization of landmarks for robotic applications. *IEEE International Conference on Robotics and Automation(ICRA 2008)*, pages 4012–4017, 2008.

[3] J. Bares and D. Wettergreen. Dante II: Technical Description, Results, and Lessons Learned. *The International Journal of Robotics Research*, 18(7):621–649, 1999.

[4] A. Baumberg. Reliable feature matching across widely separated views. *IEEE Computer Society Conference On Computer Vision And Pattern Recognition*, 1:774–781, 2000.

[5] J. Borenstein and Y. Koren. A Mobile Platform For Nursing Robots. *IEEE Transactions on Industrial Electronics*, 32(2):158–165, 1985.

[6] W. Burgard, A. Cremers, D. Fox, D. Haehnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The Interactive Museum Tour-Guide Robot. *Proceedings of The National Conference on Artificial Intelligence*, pages 11–18, 1998.

[7] B. Cartwright and T. Collett. Landmark learning in bees . *Journal of Comparative Physiology: Neuroethology, Sensory, Neural and Behavioral Physiology*, 151:521–543, 1983.

[8] B. Cartwright and T. Collett. Landmark maps for honeybees. *Biological cybernetics*, 57(1-2):85–93, 1987.

[9] J. Castellanos, J. Neira, and J. Tardos. Multisensor fusion for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 17(6):908–914, 2001.

[10] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. *Proceedingsof the IEEE International Conference on Robotics and Automation*, 1985.

[11] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, 2001.

[12] D. Churchill and A. Vardy. Homing in scale space. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[13] T. Collett. Landmark Learning and Guidance in Insects. *Philosophical Transactions: Biological Sciences*, 337:295–303, 1992.

[14] J. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal on Robotics and Automation*, 1(1):31–41, 1985.

[15] A. Davison and D. Murray. Simultaneous Localization and Map-Building Using Active Vision. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, pages 865–880, 2002.

[16] E. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–270, 1959.

[17] J. Dounart and J. Meyer. Learning reactive and planning rules in a motivationally autonomousanimat. *IEEE Transactions on Systems, Man and Cybernetics*, 26(3):381–395, 1996.

[18] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, 1987.

[19] S. Ferdaus, A. Vardy, G. Mann, and R. Gosine. Comparing global measures of image similarity for use in topological localization of mobile robots. *Canadian Conference on Electrical and Computer Engineering*, pages 913–918, 2008.

[20] P. Fiorini and E. Prassler. Cleaning and Household Robots: A Technology Survey. *Autonomous Robots*, 9(3):227–235, 2000.

[21] M. Franz, B. Schölkopf, H. Mallot, and H. Bülthoff. Learning View Graphs for Robot Navigation. *Autonomous Robots*, 5(1):111–125, 1998.

[22] M. Franz, B. Schölkopf, H. Mallot, and H. Bülthoff. Where did I take that snapshot? Scene-based homing by image matching. *Biological Cybernetics*, 79(3):191–202, 1998.

[23] D. D. Fu, K. J. Hammond, and M. J. Swain. Navigation for everyday life. Technical report, University of Chicago, USA, 1996.

[24] T. Goedeme, M. Nuttin, T. Tuytelaars, and L. V. Gool. Omnidirectional Vision Based Topological Navigation. *International Journal of Computer Vision*, 74(3):219–236, 2007.

[25] T. Goedeme, T. Tuytelaars, and L. Van Gool. Fast wide baseline matching for visual navigation. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:24–29, 2004.

[26] H. Gross, A. Koenig, and S. Mueller. Omniview-based concurrent map building and localization using adaptive appearance maps. *IEEE International Conference on Systems, Man and Cybernetics*, 4:3510-3515, 2005.

[27] C. Harris and M. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, 15:50, 1988.

[28] M. Jogan and A. Leonardis. Panoramic Eigenimages for Spatial Localisation. *Proceedings of the 8th International Conference on Computer Analysis of Images and Patterns*, pages 558–567, 1999.

[29] J. Koenderink and A. v. Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55(6):367-375, 1987.

[30] S. Koenig and R. Simmons. Passive Distance Learning for Robot Navigation.

[31] S. Koenig and R. Simmons. Unsupervised learning of probabilistic models for robot navigation. *IEEE International Conference on Robotics and Automation*, 1996.

[32] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. *Proceedings of the national conference on Artificial intelligence*, pages 979–984, 1994.

[33] B. Krose, R. Bunschoten, S. Hagen, B. Terwijn, and N. Vlassis. Household robots look and learn: environment modeling and localization from an omnidirectional vision system. *IEEE Robotics & Automation Magazine*, 11(4):45–52, 2004.

[34] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2(2):129–153, 1978.

[35] B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119(1-2):191–233, 2000.

[36] B. Kuipers and Y. Byun. A robust qualitative method for robot spatial learning. *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, 88:774–779, 1988.

[37] B. Kuipers and Y. Byun. A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations. *Robotics and Autonomous Systems*, 8:47–63, 1991.

[38] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[39] V. Lumelsky and A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, 31(11):1058–1063, 1986.

[40] M. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

[41] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Hierarchical image-based localisation for mobile robots with monte-carlo localisation. *Proceedings of European Conference on Mobile Robots (ECMR03)*, pages 13–20, 2003.

[42] Y. Mezouar, H. Abdelkader, P. Martinet, and F. Chaumette. Central catadioptric visual servoing from 3D straight lines. *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1:343–348, 2004.

[43] H. Moravec. Certainty grids for sensor fusion in mobile robots. *Sensor Devices and Systems for Robotics*, pages 243–276, 1989.

[44] T. Pajdla and V. Hlavác. Zero Phase Representation of Panoramic Images for Image Vased Localization. *Proceedings of the 8th International Conference on Computer Analysis of Images and Patterns*, pages 550–557, 1999.

[45] J. Porta and B. Krose. Appearance-based concurrent map building and localization using a multi-hypotheses tracker. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, 4:3424–3429, 2004.

[46] J. Puzicha, T. Hofmann, and J. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. *Computer Vision and Pattern Recognition*, pages 267–272, 1997.

[47] Y. Rubner, C. Tomasi, and L. Guibas. The Earth Movers Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[48] P. Rybski, S. Roumeliotis, M. Gini, and N. Papanikolopoulos. Appearance-based minimalistic metric SLAM. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 1:194–199, 2003.

[49] C. Schmid and R. Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, pages 530–535, 1997.

[50] B. Scholkopf and H. Mallot. View-Based Cognitive Mapping and Path Planning. *Adaptive Behavior*, 3(3):311–348, 1995.

[51] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 2:2051–2058, 2001.

[52] H. Shatkay and L. Kaelbling. Learning Topological Maps with Weak Local Odometric Information. *International Joint Conference On Artificial Intelligence*, 15:920–929, 1997.

[53] D. Stricker, P. Daehne, F. Seibert, I. Christou, L. Almeida, R. Carlucci, and N. Ioannidis. Design and development issues for archeoguide: An augmented reality based cultural heritage on-site guide. *International Conference on Augmented, Virtual Environments and 3D Imaging*, pages 1–5, 2001.

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

[55] S. Thrun. A Probabilistic On-Line Mapping Algorithm for Teams of Mobile Robots. *The International Journal of Robotics Research*, 20(5):335, 2001.

[56] S. Thrun and A. Bucken. Integrating grid-based and topological maps for mobile robot navigation. *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 2:944–950, 1996.

[57] S. Thrun, W. Burgard, and D. Fox. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Autonomous Robots*, 5(3-4):253–271, 1998.

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

[59] S. Thrun, D. Hahnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. *Proceedings of the IEEE International Conference on Robotics and Automation*, 3:4270–4275, 2003.

[60] E. Tolman. Cognitive maps in rats and men. *The Psychological Review*, 55(4):189–208, 1948.

[61] S. Tully, H. Moon, D. Morales, G. Kantor, and H. Choset. Hybrid localization using the hierarchical atlas. *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2857–2864, 2007.

[62] I. Ulrich and I. Nourbakhsh. Appearance-Based Place Recognition for Topological Localization. *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1023–1029, 2000.

[63] A. Vardy. A simple visual compass with learned pixel weights. *21st Canadian Conference on Electrical and Computer Engineering*, pages 581–586, 2008.

[64] A. Vardy and R. Möller. Biologically plausible visual homing methods based on optical flow techniques. *Connection Science*, 17(1):47–89, 2005.

[65] R. Wehner. Visual navigation in insects: coupling of egocentric and geocentric information. *Journal of Experimental Biology*, 199(1):129–140, 1996.

[66] N. Winters. *A Holistic Approach to Mobile Robot Navigation using Omnidirectional Vision.* PhD thesis, University of Dublin, Trinity College, 2001.

[67] Y. Yagi, S. Fujimura, and M. Yachida. Route representation for mobile robot navigation by omnidirectionalroute panorama Fourier transformation. *Proceedings of IEEE International Conference on Robotics and Automation*, 2:1250–1255, 1998.

[68] J. Zeil, M. Hofmann, and J. Chahl. Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469, 2003.