Issues in Autonomous Mobile Robot Navigation

Amit Singhal [1]

Computer Science Department
University of Rochester

May 5, 1997

[1]The author would like to express his gratitude to everybody who read preliminary versions of this report and gave invaluable feedback; especially Chris Brown, who spent innumerable hours poring over this paper. It would not have been possible without him.

Abstract

Three main problems facing outdoor autonomous mobile robot navigation are unstructured environments, moving obstacles, and multiple sensors. Each of these leads to uncertainties that usually cannot be resolved using techniques popular for indoor navigation. However, by modularizing the task of navigation and treating it as three different subtasks (robot localization, goal recognition, and path planning), we can use some of the techniques in combination to build a mobile robot system than can work in the outdoor world. We present and summarize an extant body of literature dealing withthe unstructured dynamic environments observed via multiple sensing modalities. An in-depth survey of the current research literature in the field shows the lack of research efforts directed at generating generalized autonomous mobile robot navigation systems.

We believe that while a single technique cannot be used to create a robust and efficient generalized autonomous mobile robot navigation system, a combination of some of the different methods in a probabilistic framework can achieve this goal. We propose a probabilistic framework that is based on the generalized framework approach. We use a modified version of the occupancy grids, called *dynamic occupancy grids* as our representation of the environment and a modification of Bayesian networks, termed *dynamic Bayesian networks* as our tool for updating the dynamic occupancy grid representation.

We use an asynchronous update methodology that uses dynamic Bayesian networks to create new probability density functions (PDF) that are then used to update the information stored in the occupancy grids. Whenever a robot needs to perform some new motion task, it discretizes the PDFs currently associated with the dynamic occupancy grids to create a unified *view* of the current state of the environment.

We have recently acquired two mobile wheelchair robots with limited *off-road* navigation capabilities. These will serve as a test-bed for our framework in the real world. We are also in the process of building simulated virtual environments on an SGI machine for use as a simulator test-bed to model multiple complex test environments. We are currently involved in two major research projects related to the development of our proposed framework. The first research effort was performed as part of the Mobile Robot practicum offered in the Spring semester of 1997 and is aimed at developing and implementing an obstacle avoidance system for the wheelchair robots. The second research effort is aimed at developing an efficient, platform independent parallel implementation of the memory-based object recognition algorithm proposed by Nelson. We describe our recent successes and failures in both of these efforts and highlight some of our future research plans.

Contents

Introduction

Gone are the days when rickety buckets of tin and steel were operated by remote users and passed off as robots. Today's high performance world demands precise and scientific solutions from the robotics industry. The newer robots are more mobile, carry larger payloads, perform a varied number of tasks, are capable of learning new operations, and most importantly, do all this *independently* of a human controller [6], [36], [57], [74], [77].

However, to claim that we have robots that truly exhibit even a majority of the above mentioned traits would be an outright lie. Mobile autonomous robots are very much in the research and development phase. Even though there have been a few commercially viable robot systems created in the past decade [2], [6], a general purpose mobile robot that truly exhibits the intelligence required to learn new tasks and perform them in an autonomous manner eludes us. What are some of the problems that have prevented us from achieving a robot that looks and functions like the famed android *Data* from the popular *Star Trek* television series?

Let us forget about the external physical characteristics for a while. That is a problem most computer scientists would rather leave for the adventurous biologists amongst us. The biggest obstacle preventing the robotics community from creating artificially intelligent mechanical beings is the enormity of the human thinking, learning and acting processes that need to be modeled.

One can say that robotics, like computer science, is fundamentally about algorithms [47]. However, robots are much more versatile mechanically than computers. They are equipped with sensors (providing *observability*) and actuators (providing *controllability*), forcing interaction with the system, and not just passive *data compliance*. Traditional computational algorithms used for problem solving in computer science cannot apply in this domain. This led Latombe to propose that robot algorithms be viewed as a blend of basic control issues (controllability and observability) and computational issues (calculability and complexity) [46].

From this perspective, we can classify research in artificially intelligent robotics into three main branches. The first branch attempts to gain cognizance of the brain's intelligent processes (learning, thinking, reasoning etc.). The second branch focuses on providing observability and controllability via sensor modeling, data fusion, control theory and related topics. The third branch comprises efforts to create tractable algorithms to solve the problems identified in the first two branches.

This report presents an analysis of research activity spanning all three branches of intelligent robotics in an effort to acquaint the reader with the broader issues, efforts and successes in *autonomous mobile robot navigation*. We hope that by the end of the report, the reader will be able to appreciate the complex issues being tackled by colleagues in other fields that have direct and indirect implications in their

own field of research. A unified and coherent view may be able to help the readers understand some of the issues in their own area in a broader context, and spawn future inter-disciplinary research efforts to a greater extent.

The majority of past and current research work focuses on navigation in structured, indoor environments. Only recently have people started to experiment with methods for autonomous navigation in outdoor environments. What has been done in this area before? How well has it worked? What can be done to improve these results? Chapter 1 of this report presents the autonomous mobile robot navigation problem and its characteristics. An extensive body of literature is summarized in the second chapter. The third chapter presents a generalized framework and methodology we propose to help improve the performance of autonomous mobile robots. The fourth chapter highlights some of our current experimental research efforts and preliminary results. The final chapter attempts to link the previous chapters into a coherent framework and strategy for the future.

Chapter 1

Autonomous Mobile Robot Navigation

1.1 Introduction to the Problem

Autonomous mobile robot navigation attempts to provide a means for an autonomous robot to move safely from one location to another. In this section we discuss some of the issues that complicate this problem and highlight the past and current methodologies favored among the research community.

The biggest constraint facing research efforts in intelligent robotics is that of computation power. When compared to the computational power of the brain, we have very limited computational ability. This limitation prevents us from achieving real time performance, even if we had complete understanding and knowledge of the brain's intelligent processes. Other factors that further complicate outdoor navigation algorithms include the object recognition problem, the presence of (possibly random moving) obstacles, the uncertainties generated by multiple sensing modalities used by a robot for observation, and the weaknesses of sensing devices themselves along with poor theory and control of sensors [18].

1.1.1 Computational Power

Though we keep building faster CPUs and cheaper memory, most computer scientists believe that we will soon reach a plateau in the exponential growth of computational power witnessed in the last decade [23]. The processing power required for real time image processors, computer vision, biomedical systems, telecommunications and machine learning far exceeds that of present day supercomputers. CMOS technology has almost reached its limits and any new hardware design paradigm (possibly DNA

computing [37] or Photonics [33]) that may provide the computational resources we need is still years away from being commercially viable [23].

The computational limitation notwithstanding, there exists a plethora of tractable implementations of the problems posed by researchers in *neural modeling*, *sensor modeling*, *data fusion* and *control theory*. Apostolopoulos *et al.* describe a vision guided autonomous floor-tile laying mobile robot that possesses integrated sensor and actuator control [2]. *Blanche* is a mobile robot designed for structured office or factory environments and uses an off-line path planner to generate collision-free maneuvers with the additional assumption that the vehicle's location is always known [16]. CMU's *NAVLAB* project has made major national headlines with its first completely autonomous cross country trip. It uses computer vision algorithms to detect road edges, lane markers and other vehicles on the road to perform collision free driving [36]. As a result of the above research successes, practical mobile robots now exist.

1.1.2 Object and Landmark Recognition

The common thread amongst all of the research highlighted in the previous paragraph is the assumption on the *visual structure* of the environment. Although NAVLAB operates in the outdoor environment, it relies on edge detection capabilities afforded to it by the physical structure of the roads it drives on. The other two robots operate strictly in a linearly structured indoor environment with predominantly straight-line image featurs, with no obvious extension to an *unstructured* (lacking simple geometric shape or visual features) outdoor environment. Indoor environments exhibit a high degree of deterministic linear and non-linear visual features, providing robots with good structure and organization for image processing tasks. Outdoor environments tend to be unstructured and *defeat* computer vision solutions developed for the former [18].

How can one approach the problem of navigation in unstructured outdoor terrain? Most of the commonly used approaches to landmark and object recognition fail in the absence of linear visual features. Nelson's [61] memory-based object recognition algorithm recognizes objects with non-linear visual features but requires a training phase where precise multiple (around a sphere) views of each object must be stored in a database. Acquiring these views can be a very hard task without accurate small scale models of every object in the environment. Another potential problem is the amount of time required to perform recognition on large databases of objects. Despite these drawbacks, it is one of the best object recognition systems for use in unstructured domains because of its ability to detect classes of objects based on a learned generalized model. We discuss this system and some current research work aimed at improving its performance in detail in the following chapters.

1.1.3 Obstacle Avoidance

Another source of uncertainty in mobile robot navigation is the presence of moving obstacles. While the behavior of some of these (eg. car driving down the road) can be characterized and predicted, other objects (eg. child running across a field) may exhibit randomness in their motion. *Path planning* refers to generating a route from one location in the environment to another, and path planning algorithms attempt to generate a collision free motion trajectory for the robot over a time period. Path-planning is known to be an NP-hard problem in a dynamic environment [67]. Another result shows that the above problem is PSPACE-hard even in the presence of constraints such as velocity bounds [13].

Despite the hardness of the path planning problem, various methods have been proposed to attack the problem in a dynamic environment. The first approach relies on a *space-time* formulation. Given the predicted motion of all the obstacles in the environment, a *reachability region* containing all possible locations for a robot in some future time is created. This is then used to guide the robot to a desired location [26], [40], [79]. The major problem with this approach is the amount of time required to construct the reachability region. To reduce the amount of computation time involved, Suzuki and Arimoto proposed a divide and conquer strategy for solving the above problem by decomposing it into subgoals that could be solved easily and then concatenated to obtain the final plan [73].

The above approach is extremely constrained by the precise demand of prior knowledge about all obstacle motions and trajectories. A second group of approaches has been proposed to deal with unexpected obstacles and unknown trajectories. This approach, commonly called *collision avoidance* or *collision detection*, is based on *reactive navigation* and is not subject to the hardness constraints imposed on the planning problem [1], [7]. While most reactive navigation systems tend to be based on ad-hoc designs, they can also be learned from observations made of the environment [7], [80], [77]. While neither of the above require any prior knowledge of the obstacle motion, they have their drawbacks. Learning can be a very time consuming process, and has to be repeated whenever there is a change in domain constraints. Fuzzy control can be very *ad-hoc* and relies on expert knowledge that may be flawed. Another drawback with both of these newer approaches is that they do not involve any aspect of planning and are purely reactive. One can usually derive some partial information about obstacle motion over a series of observations and use it in an advantageous manner to reduce the complexity of the obstacle avoidance algorithm and increase its reliability.

1.1.4 Multi-Modal Sensor Fusion

The third factor contributing to uncertainty in the mobile robot navigation problem is conflicts and inconsistencies between multiple sensors on the robot. Autonomous robots rely on numerous sensors to obtain a consistent and coherent view of the current world state. This inherently introduces uncertainties as different sensors may react differently to the same stimuli, or may provide incorrect or inconsistent data. These sensor discrepancies have to be handled in some framework to allow the robot to *visualize* a unified view of its environment. Most advances in *sensor fusion* have largely entailed rediscovery and adaptation of techniques from estimation theory [17].

The early work characterized the sensor fusion problem as one of incremental combination of geometric information. Most of these techniques were *ad-hoc* and did not characterize the uncertainties in the system. The first work on incorporating uncertainty in an explicit manner in sensor fusion was performed by Smith and Cheeseman [70]. They proposed the use of *Bayesian estimation theory* and derived a combination function that was an equivalent form of *Kalman Filter*. This caused a rapid paradigm shift towards probabilistic estimation theories closely related to *Bayesian estimation, maximum likelihood estimation* and *least squares methods* [17]. While these methods have been successful in combining sensor data that determine a common subset of the robot state vector (eg. robot location derived from GPS, odometry and gyroscope readings), there seems to be no obvious extension to *multi-modal* sensor fusion processes. The advantage of multi-modal approaches is that one can combine data from dissimilar sensors (eg. decibel-meter providing noise levels and camera providing visual imagery) to mimic human-like reasoning and guidance strategies (eg. hearing a car implies a higher likelihood of seeing a car).

In the following chapters, we discuss some of these techniques in further detail and propose a modified framework that provides us with an easy methodology for characterizing such multi-modal reasoning strategies.

1.2 A Modularized Approach

Given the uncertainties resulting from the above three issues, it is not easy for a mobile robot to move from one location to another autonomously. However, a modular approach can help reduce the complexity of the navigation problem by tackling smaller portions of it somewhat independently of each other and then combining the solutions. Leonard and Durrant-Whyte [48] summarized the general problem of autonomous mobile robot navigation using three questions : "Where am I?", "Where am I going?", and "How do I get there?". The first problem is commonly known as *robot localization*, the second as *goal recognition*, and the third as *path planning*. Each of these three

problems can be (and has been) studied independently of each other. Most researchers attempt to solve just one aspect of the autonomous mobile robot navigation problem rather than all three.

1.2.1 Robot Localization

In order to solve the localization problem, a robot needs to poll its sensors and combine these observations with a-priori knowledge of the environment to estimate its position. This a-priori knowledge can exist in the form of geometric or topological maps that may have been learned by the robot during an *exploration* phase or provided to it during the *initialization* phase. A-priori knowledge can also exist in terms of the *cause-effect* relationships between various entities in the world. As an example of the latter type of knowledge, the presence of high noise levels in a factory may be associated with regions that have heavy machinery operating in them. If there are multiple locations with similar visual cues, this kind of associative knowledge can help the robot distinguish between a working machine and a broken machine.

Perhaps the most interesting result of a survey of the vast body of literature on robot localization is that no generalized elegant solution to the problem exists. The many different methodologies proposed can be divided into the following two groups [6] : *relative* and *absolute* position measurements. Most implementations of localization algorithms in current use combine multiple approaches from the two categories.

Relative Position Methods

Relative position methods rely on two approaches : *odometry* and *inertial navigation.* Odometry uses wheel encoders recording the revolutions of each wheel to measure the distance and the direction of robot motion. The major advantage of this approach is that it can always provide an estimate of the new position based on the telemetry received from the encoders. The drawback is that the position error grows boundlessly due to slippage and drift unless an independent reference is used periodically to reduce the error [16].

Inertial navigation employs gyroscopes and accelerometers to continuously measure minute accelerations along the three directional axes. The velocity and position are estimated by integrating the measurements over time (once to obtain velocity and twice to obtain position) [6]. The major advantage of inertial navigation is that, like odometry, it is completely self contained. However, it cannot be used for an extended period of time as any positioning error increases without bound after integration, due to imprecision and drift. Highly accurate gyroscopes in current use are also prohibitively expensive, though newer products may bring this cost down [24].

Absolute Position Methods

Some of the commonly used absolute position measurement methods include *triangulation of active beacons*, *trilateration of active beacons*, *artificial landmark recognition*, *natural landmark recognition*, and *model matching*.

In the first method (triangulation), the direction of incidence of three or more actively transmitted beacons is measured and used to determine the absolute position and the orientation of the robot (see Appendix A for details). In the second method (trilateration), the distances to three or more actively transmitted beacons is measured and used to determine robot position and orientation. The calculations for trilateration are almost similar to those for triangulation. *Global Positioning Systems* use trilateration techniques based on time of flight information for uniquely coded radio signals. Distances from the robot to at least three satellites (beacons) are required to determine latitude, longitude and elevation [6].

For both methods, the only a-priori information required is the exact location of each of the active beacons. One problem with these methods is that the beacons need to be extremely powerful to ensure omni-directional transmission over large distances. This is not very practical, making it necessary to focus the beacon within a cone shaped propagation pattern. This can result in the beacons not being *visible* in some areas. Another drawback is that the radio signals transmitted by these active beacons are susceptible to distortion from refraction and reflection due to atmospheric and geographic conditions, giving a different measurement than that associated with the optimal straight line path [42].

In *artificial landmark recognition*, uniquely determinable and specially designed markers or objects are placed in various a-priori known locations in the environment for the sole purpose of enabling robot navigation [75]. The advantage of such a method is that the robot location can be uniquely determined even in adverse conditions. Triangulation can also be performed (when three or more landmarks are visible) to get exact positions. The biggest disadvantage of this approach is that it can be very inaccurate when the robot is further away from the landmark. A higher degree of accuracy can only be obtained over very short distances. Substantially more processing is necessary than with active beacon systems. Other problems include lack of recognition due to ambient lighting, marginal visibility, and competition from other objects in the environment with similar features. Another disadvantage is that artificial landmark recognition requires careful engineering of the environment, something that may not always be possible [6].

Natural landmark recognition attempts to overcome the last disadvantage of the above method by trying to recognize distinctive features that are an inherent part of the environment. The environment must be known in advance but does not have to be engineered in any manner. Some added disadvantages include increased computation

time due to more complex object recognition tasks and decreased reliability [29], [75]. This is especially true if one uses natural landmarks in an outdoor environment since the time required to recognize non-geometric and arbitrarily shaped landmarks (typically present in outdoor environments) is much higher than that required for recognizing linear landmarks (peg. doors and windows in the indoor environment) [6]. Landmark navigation approaches that rely on fast recognition of a limited number of landmarks (eg. doors and windows) require a series of observations before a unique position can be determined. On the flip side of the coin, one can increase the number of natural landmarks enabling the robot to uniquely determine its location after each observation. However, increasing the number of objects that need to be recognized increases the time required for recognition [29]. Thrun proposes a learning technique where the robot determines a balance between the two opposite approaches mentioned above [75].

Model matching, also known as *map based positioning*, involves matching the robot sensor outputs to a map or model of the environment. This technique is often also employed to update global maps in a dynamic environment, and to build a global map from multiple local maps. The two most often used map representations are topological and geometric. Geometric maps represent the world in a global coordinate system, while topological maps represent the world as a network of nodes and arcs [74]. Cruise missiles used a *terrain correlation* technique for model matching based navigation before the advent of military GPS [63]. Disadvantages of map based positioning systems arise from the specific requirements for satisfactory navigation. There need to be enough stationary, easily distinguishable features that can be used for matching. The sensor map and the terrain maps must be accurate enough to be useful. Also, correlation (the most common matching technique) requires a significant amount of sensing and processing power to be computationally feasible [6].

1.2.2 Goal Specification and Goal Recognition

Strictly speaking, *goal specification* and *goal recognition* are two different problems. Goal specification involves identifying the goals to the mobile robot navigation system. This is usually a task of the human designer. Goal recognition is then performed by the robot and refers to identifying current and future goals of the robot based on sensor observations and goal specifications. Goal recognition is an essential ingredient of the *path planning* problem described in section 1.2.3. Once the goals of a robot have been identified, a path that achieves these goals can be generated and executed via corresponding control commands to the proper hardware. Goal specification and goal recognition are so inter-dependent that we choose to discuss them together. There are many different approaches to the goal specification and goal recognition problems.

The first method of goal specifications simply uses an a-priori known map of

the environment and specifies the goal location in terms of absolute coordinates in the map. A goal can also be specified in terms of relative coordinates from the current robot location or some other landmark in the environment. In this case, goal recognition deals with converting this relative location to an absolute location in the environment.

Another approach utilizes beacons or landmarks at known locations in the environment to specify the goals. These goals can be a-priori known to the robot, reducing the goal recognition task to an object recognition task or a landmark recognition task. An example of this approach is the use of active transmitting beacons that serve as *homing* devices for the robots. While this approach is easy to use, it may require engineering of the environment and can only be used when the locations of the objects or landmarks are known a-priori and the robot is given an accurate map [25].

An extension to the above approach attempts to use *user-friendly goal specifications* to identify the goals to a robot [69]. The position of the goal is not given by its coordinates but by a verbal description of the route to it. Thus one can specify a goal by a series of statements such as " turn right and go to the second intersection, and...". Here, distinctive places such as intersections serve as *landmarks* in a broad sense. The robot navigation system has to take these specifications and convert them into goal locations, so that a path to these locations can be planned. Since multiple instances of each landmark may exist, the navigation system keeps track of previously identified landmarks and their locations to disambiguate. The major advantage of this approach is that accurate a-priori knowledge about the locations of the goals is not required. It also breaks the path planning problem into easily identifiable components, (each statement representing a subgoal), reducing the complexity of motion planning and making it tractable [69].

Another extension to the above attempts to deal with unknown environments by identifying goals (landmarks or objects) without specifying their locations or a path to them. This is the reverse of the very first approach in that there is minimal goal specification and completion of the task depends heavily on goal recognition. The robot then employs a reactive navigation approach to determine when one of the goals is *visible* and then plans a path to it. Object or landmark recognition algorithms can be used to constantly monitor visual input for the specified landmarks. This approach can be further extended to *learn* the environment during the initial explorations and trials. This *learned* partial information can then be used by the goal recognition algorithm to guide the robot when one of the known landmarks is specified as a goal [19].

The above approaches to goal specification and recognition identify a goal as a landmark or location that the robot is aiming to reach. A goal can also be something more abstract and *task directed*, eg. pick up all the balls from the playing field. In this approach, the robot has to constantly poll its sensors to find the next goal state (the

next ball to pick up) until it has picked up all the balls. The *subsumption architecture* was the first attempt at a generalized formalization of this approach [11].

In the subsumption architecture, each robot is associated with a repertoire of *behaviors* that it can exhibit. Goal recognition is performed via polling the environment and selecting the next goal (and the corresponding behavior that achieves this goal). Any change in the environment can cause the currently executing behavior to be subsumed by a new behavior (eg. a deer wandering through the forest may be searching for food but seeing a hunter will cause it to hide in a cave). The subsumed behavior may be resumed at a later time [11].

1.2.3 Path Planning

Once the goals of a robot have been identified, the next task is to navigate towards them. The two major motion strategies used for mobile robot navigation are *reactive* and *planning* [27].

In reactive navigation, only the goal is known (and sometimes its location). However, no explicit encoding of a path that can be followed to achieve that goal is generated. The robot periodically polls its sensors to determine the current state of its location and environment. The sampling rate can vary with the degree of uncertainty (arising due to clutter, mobile obstacles, ambiguities) associated with the environment. Based on these observations, a robot can change its speed and orientation, avoid obstacles, or perform some other task. Reactive navigation works especially well in the presence of dynamic obstacles and in unknown environments [19]. It has the added advantage of high flexibility, although it can be less efficient than a path planning strategy as it may take a long time to achieve the goal based purely on sampling the environment state. Another problem with reactive navigation systems is that they may reach deadlock states or get stuck in cycles without ever progressing towards the goal.

Planning strategies attempt to generate an *optimal path* (shortest collision-free path) from the current location to the desired location given a-priori knowledge of the world (eg. as a map). Again, the robot polls its sensors (though with a much lower sampling rate than in the reactive case) and determines its location in the environment. Once this location is known, the next goal (or sub-goal) is recognized and a path to it generated. Ideally, the robot does not need to deviate from this path since there exists extensive internal modeling of the environment in the form of maps that capture the location of all obstacle and specify other constraints. However, uncertainties can be introduced into this internal representation by the presence of moving obstacles, multiple conflicting sensors etc. [31].

To take advantage of the properties of the two approaches, most mobile robots use a combination of reactive and planning navigation. Usually, once the robot has

determined its location and its new goal, a path is generated. The robot follows this path but periodically surveys its sensors to determine the presence of moving obstacles or other constraints that may require it to deviate from the generated path. It then uses reactive navigation to move around the obstacle and get back on the original path, from where the path-following algorithm can take over once again. Some newer approaches also predict the path of moving obstacles and account for that in the path generated for the robot [77].

1.3 Motivating a Survey

The previous sections were aimed at providing the reader with a high level description of some of the issues facing autonomous mobile robot navigation, and the attempts made at generating solutions for these. We identified the problems associated with mobile robot navigation, especially those that introduce uncertainty into the system. Most early solutions were deterministic in nature and based on assumptions that became invalid in the presence of mobile obstacles, multiple conflicting sensors, and other uncertainties associated with outdoor and dynamic worlds. We also presented a cursory history of the progress made in the field and a general description of some of the techniques that have found favor among the research community.

In the next chapter, we present an in-depth analysis of some of the most successful and recent attempts at solving the more generalized problem of autonomous mobile robot navigation in an uncertain environment. We plan to use the literature survey to motivate the need for further work in generalized mobile robot navigation and propose a new framework that utilizes some of the existing techniques to unify various aspects of the mobile robot navigation problem.

Chapter 2

A Literature Survey

2.1 Introduction

As discussed in the previous chapter, three main problems facing outdoor autonomous mobile robot navigation are unstructured environments, moving obstacles, and multiple sensors. Each of these leads to uncertainties that usually cannot be resolved using techniques popular for indoor navigation. However, by modularizing the task of navigation and treating it as three different subtasks (robot localization, goal recognition, and path planning), we can use some of the techniques in combination to build a mobile robot system than can work in the outdoor world. This section presents and summarizes an extant body of literature dealing with the unstructured dynamic environments observed via multiple sensing modalities.

We start with a general overview of the state of the art in outdoor and indoor mobile robot navigation and present some techniques that have been successfully applied to solving the navigation problem in structured worlds. Next we present research work done in identifying artificial and natural landmarks in outdoor environments and some object recognition algorithms. Since there may be moving obstacles present in an environment, we study this literature and summarize some of the methods used to detect and avoid moving obstacles. Also, a robot may be equipped with multiple sensors that must provide it with a single consistent and coherent view of the world at any given time. We describe some of the methods used for such multi-sensor fusion and integration. Finally, we study some probabilistic techniques that allow us to deal with uncertainties in the world.

2.2 The State-of-the-Art

Autonomous mobile robot navigation has made much progress in the last decade. Though a single elegant solution to the navigation problem still eludes us, various combinations of the many different methods proposed have provided good results, both in simulation and in the real world. However, most of these highly successful endeavors have been in highly structured and static environments.

Apostolopoulos et al. describe an autonomous mobile robot for floor tile laying [2]. The robot is capable of omni-directional locomotion and uses stereo cameras and light-striping for external sensing. It is also equipped with an internal dead-reckoning system based on its wheel encoders. A sample trial of the robot consists of laying a floor of tiles in a simple grid pattern. It is assumed that the edges of the tiles can be easily detected using a combination of the Canny operator and the Hough Transform. A non-contact stereo camera based technique is used to locate tile edges and seams on the floor to place subsequent tiles. Light striping is used to verify the alignment of the tiles. The robot is equipped with mechanical devices required to lay the tile and has an average speed of approximately 12 seconds per tile (compared to almost 24 seconds per tile for humans). Some limitations of the robot include a human-assisted start up phase for deployment, facility sizing, tiling-job planning and initiation of tile placement sequence. The robot is also incapable of laying odd and cut tiles and creating intricate tile patterns. Some other partially successful attempts at autonomous indoor mobile robot navigation are described in greater detail in [14], [51], [81].

The presence of mobile obstacles, unstructured terrain and multiple (possibly) dependent sensors introduce uncertainties and complexities that pose a much harder problem than navigation in a static structured domain [6], [44]. Most of the current research work is targeted towards finding better solutions to the autonomous navigation problem in less and less constrained environments. CMU's *NAVLAB* project [36] and Dickmanns' *VaMoRs* project [18] lead the autonomous *on-road* driving research efforts in the world. Both use edge detection algorithms to track roads and driving lanes. Object recognition algorithms are used for detecting obstacles and other cars and localization is done using road features like intersections, lane changes, corners, and bridges as landmarks. Recent extensions to these systems have also added localization using a GPS receiver. NAVLAB's road tracker is also capable of detecting dirt roads and furrows for limited rough terrain navigation. Recently, a host of papers on unmanned aerial and underwater vehicles have also come to the forefront of autonomous vehicle research, foremost among them being the *Ocean Explorer* and *Cruise Missile* projects [57]. Ocean Explorer uses a topographical map and model-matching techniques to perform underwater navigation. Cruise missiles originally used a similar model-matching approach but now use a military-grade GPS system

for navigation. Though neither of the two are fully autonomous vehicles, they exhibit moderate intelligence in that they are capable of sensing the environment and changing their planned paths of motion.

Autonomous off-road mobile vehicles pose certain unique challenges that defeat most on-road and indoor robot navigation algorithms. Lack of accurate and detailed maps, potentially hazardous terrain and dynamic objects that may change the environment make safe and reliable navigation hard. Some of the current areas of focused research include *robotic control*, *sensing* and *path planning* (both in the presence and absence of a-priori knowledge of the environment) [57].

Some of the key techniques being studied for robot control include classical open loop control and closed loop feedback control, potential field control [28], behavior based subsumption control [56], model-and-planner approach [49] and various combinations and adaptations of these. Mataric uses a behavior based approach to perform cooperative tasks with multiple robots. This approach differs from purely reactive control in that there isn't just a simple functional mapping between stimuli and appropriate responses, but also a distributed aspect wherein multiple behaviors may be active at any given time and a central policy decision chooses one of these to execute [56].

Multi-sensor fusion and integration research is a two pronged effort, one focusing on representations (eg. occupancy grids [21]) and the other focusing on methodologies (eg. Bayesian networks [43]). At the same time, physicists and electrical engineers are also involved in inventing new sensor technologies, the most notable of the recent ones being GPS [34]. *Computational sensors* is another very new and active sensor research area. Here, raw sensing and computation are integrated together on the sensor itself, eliminating the need to transmit sensor information to some other computational resource for processing [9].

Most of the computational sensor research to date has been limited to local sensory data preprocessing (eg. pixel-wise adaptation and convolution with a small kernel). This does not reduce the sensor-processor data transfer bottleneck by any appreciable amount, resulting in continued excessive latencies. Researchers at CMU (Brajovic and Kanade) are at the forefront of computational sensor research and are working on parallel computational sensor models that are capable of making global decisions about sensor observations. Two such sensors are the *winner-take-all sensor* (WTA) and the *sorting sensor*. The WTA tracking sensor detects the local peak in the image intensity and continuously tracks it. The sorting sensor computes a histogram of the image intensities and assigns an ordinal number to each input in the histogram. With limited guidance from the user, it can perform simple operations such as segmentation and labeling [10].

The evolution of new sensors, some using completely different paradigms, creates a constant flux in multi-sensor fusion and integration methodologies. Some commonly

used approaches to sensor fusion include *Bayesian estimation* [66], *Kalman filter* and its variants [55], *fuzzy rule based* systems [78], *decision theory* [45], *neural networks* [15] and *Bayesian networks* [43]. Even though we have so many different techniques for performing sensor data fusion, we still lack a framework that can combine sensor data from multiple sensing modalities effectively. Each of the above methods works only within their defined constraints and underlying assumptions. Most of these assumptions and constraints break in the face of emerging technology. We will provide more details of these methodologies in following sections, but for now we state our objective as "a framework that can perform multi-sensor fusion from multiple sensing modalities with minimal constraints and assumptions on the environment". In the next chapter, we propose a framework that may be able to achieve this objective.

Performing path planning requires that the robot be able to perform localization first. While exhaustive research has been done in robot localization algorithms, we have not yet found a methodology that works well in minimally constrained environments. Most outdoor robots use some sort of structured (furrows along a dirt path, road boundaries, traffic signs, buildings etc.) landmarks for performing localization. These landmarks can be artificial or natural. However, their location needs to be known a-priori. GPS, an emerging technology, provides good localization even in the most adverse and unknown terrains. Its biggest disadvantage is that it provides absolute coordinates (longitude, latitude and elevation) only and cannot be used without a geometric map of the environment. Another drawback is that GPS signals cannot be received indoors [34], making GPS a highly unlikely candidate for generalized robot localization.

Current research in landmark detection algorithms is focused on detecting unstructured and arbitrary landmarks such as trees [35] and distant peaks [71], [76]. All of these implementations match the input scenes against stored topographical maps. A second approach recently proposed by Thrun views landmarks as *features learned by the robot* during an initial training phase [75]. Multiple neural networks are used as feature detectors and trained by providing a series of images from the test environment as inputs. After training, this set of feature detectors can be used to perform localization by determining the subset of features visible from various locations in the environment [75].

The majority of artificial landmark detection research can be classified into two approaches, the first based on geometric models and the second on object recognition. In the first approach, simple regularly shaped polygonal or circular patterns are used as landmarks. A perspective projection of these landmarks in a robot mounted camera's image space can then be used to compute the location estimate [51].

Object recognition algorithms have gained popularity because of the visual structure and shape associated with artificial features. There exist various algorithms that perform very well on objects with linear shapes and boundaries [30]. However, linear

constraints cannot be imposed on most environments, hence the need for recognizing objects with non-linear boundaries. Two of the most successful approaches to non-linear object recognition are those of Nayar [60] and Nelson [61]. Nayar uses *appearance representations* to encode a combined effect of the object's shape, reflectance properties, pose in the scene, and the illumination conditions into a compact model [60]. Nelson uses a feature matching technique to identify objects in a *cluttered* scene by computing the intermediate-level features (specifically 2-D boundary fragments)of objects in the input image and searching for a match in a pre-constructed database [61].

Another area of active interest is path planning in the presence of dynamic obstacles. Various approaches have been suggested to tackle this problem, ranging from *fully reactive and reflexive avoidance* [14] to *predicting obstacle motion* [59]. The first approach uses continuous sensor polling (usually proximity sensors only) to detect any obstacles that enter the robot's *personal space*. The personal space is an enclosed area surrounding the robot that should ideally be kept free of any obstacles at all times. When some object invades the robot's personal space, the robot takes evasive action and avoids the obstacle [14]. In the second approach, the robot periodically polls its sensors (typically vision and long range sensors) to detect moving obstacles and predict their trajectories. A path planner then generates a route that takes the mobile obstacles into account. The sampling rate of the sensors depends on the uncertainty associated with the prediction of the obstacle trajectories [59]. Hybrid approaches to this problem have also been studied and found favor in many implementations [8].

In the remainder of the chapter, we will look at some of this recent work in much more detail and try to give an in-depth understanding of the current state-of-the-art in these areas.

2.3 Landmark Detection

Many different methodologies for robot localization in structured indoor environments exist. These range from pure odometry or inertial navigation systems to a combination of sonar, vision and other sensing modalities for place recognition [43]. Indoor environments can usually be structured to fit the requirements of one or more of the prevalent methodologies without much hardship and, as such, no longer pose a very interesting problem to researchers. Of course, introduction of uncertainty in the form of dynamic obstacles, or minimizing the cost of navigation strategy still spawn multiple research efforts.

However, the lack of ability to structure most outdoor environments to our needs forces us to rely on a few time-tested techniques for outdoor navigation, while keeping the search on for newer methods. Artificial and natural landmark based robot

localization is one of the most popular and well studied approaches for outdoor autonomous mobile robot navigation. Artificial landmarks are structured objects specifically placed in the environment to aid localization and navigation and can be detected using image processing algorithms such as *edge-detection*, *region segmentation*, and *object recognition*. Examples of artificial landmarks include bar-codes, color coded signs, traffic and street signs, and roads.

Natural landmarks are objects or features already present in the environment and serving some other primary function besides aiding in robot localization and navigation. These can be structured (eg. buildings, lamp-posts, doors, windows, fire hydrants etc.) or unstructured (eg. trees, mountain peaks, etc.) in terms of their shape. Most natural landmark detection algorithms use structured landmarks for localization purposes as they can be detected relatively easily using a variety of image processing algorithms [55], [69]. Another approach to landmark detection employs neural networks to classify input (visual, range data, or some other sensory data) patterns and perform localization based on the classification results [68], [75]. Thompson's group at the University of Utah is at the forefront of unstructured landmark detection for robot localization. They detect mountain peaks in the input images and match their locations against a topographical map to estimate the robot position. The major drawback of using distant features for localization is the high uncertainty in the position estimate ($\pm$ 30m) [76].

We describe some of the more interesting and successful research efforts in further detail.

2.3.1 Artificial Landmark Detection

Here, we analyze some of the artificial landmark detection approaches that do not rely on object recognition methodologies. Most of these approaches classify multiple features as a *single* landmark and detect these via a variety of methods such as Hough transform [51], edge detection [18], [36], [69], and neural networks [22], [54].

Geometric Models

Simple geometric models such as circles, ellipses or polygons are often used as artificial landmarks because they can be detected easily in real time using edge detection algorithms. Circles are especially good as they project ellipses into the camera space when viewed at an angle by the robot. These projections can then be used to compute the location of the robot.

One of the best algorithms that employs this technique was recently proposed by Lin and Tummala [51]. Figure 2.1 shows the artificial 3-D landmark pattern used to perform localization. It consists of two disks (a smaller one some distance in front of

the larger one) at an a-priori known location in the environment. A robot mounted camera is used to scan a vertical stripe that spans the disk to align the camera axis with the center of the disk pattern. The modified elliptical Hough transform is then used to detect and measure the projection of this landmark pattern in the camera's image space. We can estimate the distance of the landmark from the viewpoint (the robot) using the equation :

$$D = \frac{2fR}{b}$$

Under the assumption that the distance between the robot and the landmark is relatively large compared to the dimensions of the landmark, we can compute the *aspect angle* (orientation) of the landmark with respect to the camera frame as follows:

$$\theta = cos^{-1}(\frac{a}{b})$$

where D is the distance between the center of the camera and the center of the landmark, θ is the aspect angle between the camera and the landmark, a is the length of the minor axis of the projection of the primary disk, b is the length of the major axis of the projection of the primary disk, R is the radius of the primary disk (50 cm in this case), and f is the focal length of the camera.

Disambiguity in the direction the robot is looking at the landmark is resolved by checking the location of the projection of the secondary disk (the smaller disk) with reference to the center axis of the landmark pattern (negative and positive θs).

The technique works well even in the presence of noisy data and partial occlusion (direct advantages of using the Hough transform). Another advantage is that the location estimate can be very easily computed once the distance and aspect angle between the camera and the landmark are known. The major computation involved here is performing the transformation of the input image into Hough space to detect the ellipses.

The major disadvantage of the approach is that the there is only one landmark and it must be at least partially visible to allow the robot to perform localization. This restricts the use of this approach to small environments with no walls or other obstacles that may occlude the landmark. Errors in the position estimation arise due to two reasons. First, the projection of a disk is not really elliptical but an approximation to an ellipse. Empirical results show that this can introduce an error of up to 10% in the position estimation. The second source of error is occlusion of the landmark. With at least 50% of the landmark visible, the error in the aspect angle is under 7% but degrades quickly when more of the landmark is occluded [51].

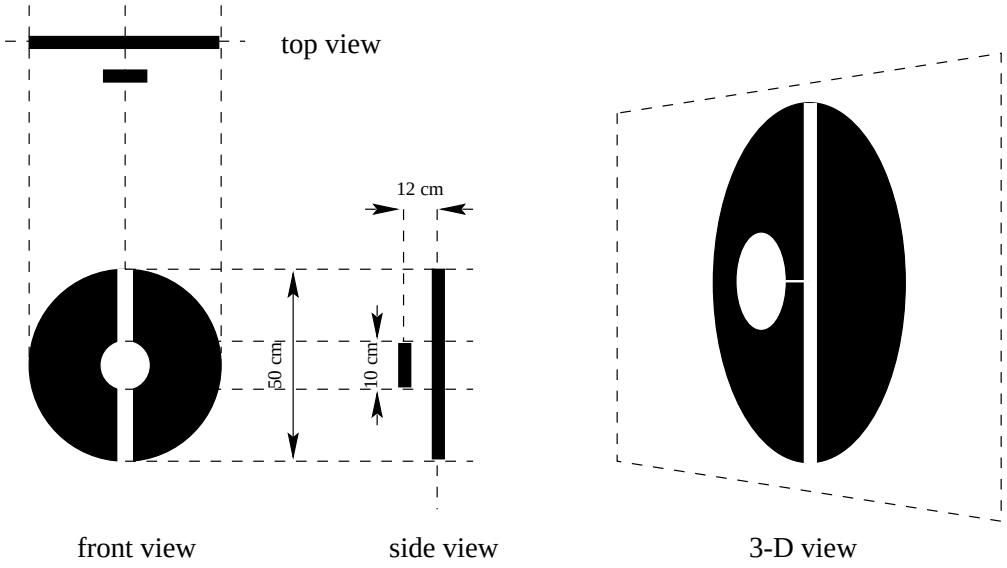

Figure 2.1: Landmark Pattern Used by Lin and Tummala

Road Tracking

Road edge detection and tracking is one of the most common navigational methods for on-road autonomous vehicles [18], [69]. The NAVLAB project has extended these road detection capabilities to allow driving on dirt paths also [36]. Localization can be performed by using intersections, corners and curved roads as landmarks and keeping track of their sequence of occurrence to periodically update the position estimate generated via odometry methods.

CMU's NAVLAB project [36] is the leader in on-road autonomous driving vehicle research in the United States. In addition to using vision based algorithms for landmark detection, it uses a neural network system to learn how to drive by simply observing the actions of a human teacher. The YARF component system of the NAVLAB tracks lane markings and other features and uses position tracking windows to detect intersections and lane changes [41]. Feature detection is done using selective application of specialized segmentation routines. As an example, intersections are detected by the failure to observe a continuous road boundary on one side of the road. ALVINN, another research effort involving the NAVLAB project uses neural networks specially trained for recognizing and driving on different road types [36].

Depending on the environment the vehicle is operating in, localization may be performed in different manners. When the vehicle is operating in a known environment, model matching techniques may be used to update the position estimation

(continuously updated by odometry) via detection of landmarks such as intersections, changes in road structure (number of lanes or curves in road), and bridges. Recently, GPS observations and traffic sign recognition algorithms have also been added to the repertoire of NAVLAB's localization methods. In unknown environments, position estimation is performed using sensor fusion from GPS and odometry. NAVLAB has been able to achieve speeds in excess of 50 mph while driving on multi-lane highways. The systems developed for the NAVLAB project work very well in outdoor environments where fine precision navigation and position estimation is not required.

VaMoRs [18], an on-road test vehicle developed by Dickmanns' research group at the University of Munich tracks road edges, lane markers and the like for performing navigation. The vehicle employs two active bifocal vision systems, one for look-ahead and the other for look-back. This allows the vehicle to maintain an internal model of the entire situation around it. The functioning of the navigation and localization systems closely resembles that of the NAVLAB systems described earlier.

Shibata *et al.* employ a different approach to performing on-road driving in unknown environments. They use user-friendly goal specifications such as "turn right at the 2nd corner, and ..." to describe the driving task instead of conventional formulations that identify the goal via some coordinates or absolute location in a map. The advantage of their system is that localization can be easily performed at such distinct locations (intersections, corners) without the need to constantly update positional estimates via odometry, GPS or other means. Positional relations among landmarks are given by the descriptions (1st intersection versus 2nd intersection) and used to disambiguate the result of each landmark recognition. An extension to this approach learns unambiguous landmark models via repeated trials through the same environment.

The biggest successes of the above systems have been in high speed driving on multi-lane paved roads. All the three vehicles described are capable of obstacle avoidance, lane changing, turning and other driving tasks. However, they are very large and complex systems composed of multiple components that cannot be easily duplicated or scaled down for other environments (such as factory delivery systems, office robots, or even outdoor robots that are small in size). Since they are built to operate in large environments, the position estimations and requirements for navigation have wider margins of absolute error that may be unacceptable in smaller environments.

Neural Networks

Another popular methodology for detecting artificial landmarks uses multiple neural networks that are trained to recognize these landmarks quickly in a sequence of input images [22], [53], [54]. Most of the commonly used neural network approaches for artificial landmark classification in the outdoor environment employ neural networks

that are trained to detect and classify road and traffic signs.

Luo *et al.* present a fractal based approach to detecting and classifying street signs and mile markers in [54]. The task of locating landmarks in a cluttered scene is very complex because the perceived size of the landmarks depends on the distance between the camera and landmark. The appearance of a landmark may also change due to variations in lighting, viewing direction, and occlusions. They propose a fractal based model that is scale, light intensity and viewing direction invariant under the assumption that each landmark has a unique texture that is encoded in the intensity image of the object. These unique textures are then modeled as fractals and proven to satisfy the invariant properties. Before the system can be used for landmark detection, a learning phase creates fractal models of the various landmarks and trains a set of neural networks to classify them correctly. Landmark detection is performed by presenting the input images to these neural networks that then identify the landmarks present in each image. While this approach has the aforementioned advantage of being invariant to scale, viewing direction and lighting changes, it possesses major limitations. Most street signs have similar structure so they are all classified as one landmark. To test the system, fractal models of very different textured objects (such as road, sky, grass, stop sign and left turn sign) were used. Obviously, this cannot be used to perform localization.

A similar approach to traffic sign recognition was proposed by Escalera *et al.* in [22]. They create a system solely targeted at detecting and classifying traffic signs. The algorithm has two main stages. In the first stage, color and shape information about traffic signs is used to segment and extract the regions of interest from input images. In the second stage, a neural network is used to classify these traffic signs.

Another approach proposed by Luo and Potlapalli in [54] uses self-organizing neural networks to detect and classify mile marker signs and cautionary signs to safely navigate the robot through its path. In order to be able to learn classification patterns for new traffic signs that may be seen (on-line learning), they propose the use of self-organizing neural networks rather than traditional multi-layer networks. No explicit feature extraction is performed on the input images, resulting in considerable savings in computation time. Instead the entire image is used for learning the patterns associated with a particular traffic sign. The learning rule used for updating of weights of the neurons is given by:

$$\mathbf{w}_k(n) = \mathbf{w}_k(n-1) - \alpha < \mathbf{w}_k.\mathbf{x}^{\perp} > \mathbf{x}^{\perp}$$

where $\mathbf{w}$ denotes a weight vector, $\mathbf{x}$ denotes an input vector (the image), $\mathbf{x}^{\perp}$ denotes a vector normal to x, and α is the learning rate. The learning rule attempts to increase the projection of $\mathbf{w}$ along $\mathbf{x}$ by reducing the projection along $\mathbf{x}^{\perp}$. It is shown to be convergent as long as $\mathbf{w}$ and $\mathbf{x}$ are not normal to each other. To handle this special

case, one can use δ perturbations in the winning neuron. The learning rule is also shown to reach an optimal solution when the learning rate is set to be a monotonically decreasing function that sums to unity over N iterations of the training phase.

An extension to the original method modified the winner-take-all rule to update weights of neighboring classifiers (units). This is useful when multiple closely related members of a class may be present. During the training phase, each unit gets associated with a class of input patterns. It then stores a measure of the distance between itself and the farthest member of the class it is associated with as a threshold. This threshold value is always less than the measure of the distance between the unit and any member of any of the other classes. If we start the network with excess number of units (more than the expected number of classes). Free units can then be used to detect and identify new classes of input patterns. The results show that the system is able to classify input signs with an accuracy of almost 92% when more than a 100 training trials are used to learn the unit classifications. This performance degrades linearly in the presence of noise and occlusions.

The major disadvantage of neural network approaches is that they require a significant amount of training before they are able to achieve good performance. Another disadvantage is that they are highly dependent on the input patterns and dynamic changes to the pattern such as occlusion and clutter quickly degrade the performance. The last method does have the advantage of being generalizable to a certain extent as it can find *average* classifications for a related set of input patterns.

2.3.2 Object Recognition

The two most popular approaches to object recognition algorithms are *appearance based* and *geometric model based*. Appearance based approaches generate and store some feature representative models of the objects that need to be recognized by the system. These model representations, commonly called databases, can be be very large [61] or very compact [60]. Recognition is usually performed by detecting *salient* features in the input image and matching these against the representations stored in the database. The salient features in the input image are dependent on the object recognition algorithm. In Nelson's object recognition algorithm, the salient features are a set of 2-D boundary fragments associated with the objects [61]. In Nayar's system, the salient feature is the projection of the input image into the eigenspace of all the training images in the database [60]. While neither of these two object recognition systems has been used for landmark detection, there is an obvious application of these algorithms to the landmark detection problem motivating us to detail these further.

Some of the other most notable work in object recognition has been done using 3D geometric models [30]. The explicit geometric model representation makes it easy

to effectively use geometric constraints and methods for easy recognition. The major drawback is that these models are not always easy to obtain. [30] cites various attempts at automatic acquisition of shape and structure geometric models using sensor data. However, none of these approaches has yielded satisfactory results and they do not lend themselves to generalizations outside their particular geometric schemata [61]. These approaches also tend to be dependent on whole image segmentation and degrade quickly in the presence of occlusions and clutter.

A different approach to road scene identification and traffic sign recognition uses structure data obtained from a *route guidance sign* (RGS) database and performs template matching against images obtained from a moving camera to recognize traffic and road signs [3]. Each object in the RGS database is modeled as a two level recursive structure. The shape of the object is modeled as a rectangle, and the characters are represented as circumscribed rectangles. Segmentation (based on intensity and color) is used to extract the signs from road scenes. Template matching operations are used to recognize the sign and the characters. Arrows are recognized using thinning and template matching operations. The drawbacks of this method include segmentation problems due to clutter and occlusion and lack of generalizability to more complex shaped landmarks.

The best candidates for a generalized object recognition algorithm seem to be the appearance based methods developed in [60] and [61]. We discuss these in more detail.

Nayar's Object Recognition Algorithm

Nayar *et al.* describe an object recognition algorithm capable of recognizing 100 complex three-dimensional objects in real time (under one second for each input image) [60]. They use the appearance based representation described in [58] to encode brightness variations in the image of an object caused by its three-dimensional space, surface reflectance properties, and illumination conditions. The appearance representation for all the objects in the database is acquired through an automatic learning phase, eliminating the need for the programmer to select the appropriate representation, design the object model and input this information into the system.

The primary task of the learning phase is to create a compact representation of a large training image set (multiple views of each object under different pose and lighting conditions). A *principal component analysis* based technique is used to perform this compression. The technique relies on using *eigenvectors* as orthogonal bases for an *eigenspace* that represents all the input images. While a large number of eigenvectors are required for accurate reconstruction of each object, a small number of them suffice to capture the significant features in each object. This reduced set of eigenvectors is then used as a basis for a lower dimension eigenspace.

A database is generated by constructing this low dimension eigenspace from all the images in the training set and then projecting each image into the eigenspace. This projection produces a set of points that can be used for constructing a pose parameterized manifold of the object using spline interpolation techniques. Any new pose of the same object presented should map close to this manifold. The database is actually a collection of points generated via dense sampling of this manifold. Object recognition is performed by projecting an input image into the eigenspace and finding the closest manifold (object identity) and the closest manifold point (pose).

The authors claim a 100% recognition rate and a 2% error in the pose estimation based on empirical studies. Also, after the system was trained, each input image was classified in less than 1 second, making the system suitable for use in real time. While these are certainly impressive results, they are obtained in a very restricted domain. Some of the assumptions constraining the system include:

- Illumination conditions are constant, thus eliminating variances caused by surface reflectance properties and shading.
- Input images can be normalized for brightness and scale to achieve invariance to sensor magnification and illumination intensity.
- The objects in the training and test images are not occluded and can be segmented from the remaining scene.

As is obvious from this list of assumptions, such a system will have degraded performance in outdoor environments where occlusions are present, illumination conditions vary easily, and clutter and background changes cause problems for the segmentation algorithms.

Nelson's Object Recognition Algorithm

Nelson [61] describes an appearance-based object recognition system that is not restricted by the assumptions constraining Nayar's work. The object recognition system uses intermediate-level features representing 2-D boundary fragments as normalized keys and descriptors for constructing and indexing the database.

The basic methodology is to represent the shape of the object by a set of boundary fragments. Under varying conditions (lighting, pose, occlusion, clutter), a boundary-detection algorithm will be able to find some of these features. Multiple 2D views of rigid 3D objects are presented to the feature extraction algorithm that extracts the boundary fragments and stores them in a database along with object and pose information. To provide robust recognition, these views are taken around a sphere

surrounding the object. Interpolation between discrete views can be performed to recognize objects at poses in between the training images.

Object recognition is performed by extracting boundary segments in an input image and thresholding the results to generate a set of *keys*. These keys represent significant boundary fragments and are used to search the database to create hypothesis for the identity and pose of all the objects that could have produced these keys. A Hough-transform like evidence combination scheme is used to update the confidence in various object hypotheses generated by this search and select the best one.

Extensive testing of the system is reported in [62]. The base case used for testing composed of a database consisting of six different objects. A recognition rate of 99% was observed with test images that were free of clutter and occlusions. In further tests, it was determined that there was a decline in the recognition rate from 99% to 97% when the number of objects in the database was increased from 6 to 24. In a second experiment, the effect of clutter on recognition rates was studied. It was determined that in the 6 object case, the recognition rate reduced from 99% to 96% in the presence of dark-field non-occluding clutter. Clutter arising from light backgrounds poses a harder segmentation problem and the recognition rate in this case degraded to 90%.

A third experiment studied the generalization properties of the system in recognizing members belonging to a class of objects (eg. airplanes, cars, and cups). The database consisted of multiple views of a subset of the members of each class. In the experimentation phase, the remaining members of the class were presented to the system for recognition. Although no conclusive results are reported, the recognition rate was in the surprisingly high range of 80-98% depending on the class.

The above results suggest that the system can be easily extended to perform object recognition for navigation purposes in a generalized environment. The major advantages of the system include robust performance in the presence of clutter or occlusions with no prior segmentation of the image required. The biggest drawback is the amount of time required to perform recognition. This varies from 30 seconds in the 6 object case to 2 minutes for the 24 object case. In order to use the system, it is necessary to reduce the amount of time required to recognize an object.

We are currently involved in a research effort to efficiently parallelize the object recognition system and hope to significantly reduce the amount of computation time. More details of this work are described in section 4.2. A second direction of continued research is aimed at implementing a verification system that checks the hypotheses generated by the system against the original input image. It is believed that this additional step in the recognition algorithm will considerably improve the recognition rate since it has been observed that even when the system misclassifies the object, the "right" candidate is generally present in the top few hypotheses.

2.3.3 Natural Landmark Detection

Most of the natural landmark detection has focused on using structured objects such as doors, windows and walls as landmarks for localization. These approaches have met with relative success in the indoor environment where such structured landmarks can be found. However, it is much harder to discover structured landmarks in outdoor environments. Here we present some of the most interesting ideas for natural landmark detection with application to outdoor environments.

Landmark Learning Approach

Many researchers are of the view that selection of landmarks used by a robot for localization should be determined by the robot itself [29], [75]. Humans perceive the world differently from robots, and structures that make themselves readily available to us for landmark based localization may not be easily detected by robots. In the traditional approach, a human designer selects the set of landmarks to be used for navigation and localization and then either hard-codes the recognition routines, or provides target labels for supervised learning. The major disadvantages of this approach include:

- The human designer must be fully knowledgeable about the characteristics of the robots sensors and the operating environment
- The human sensory apparatus differs from that of mobile robots, thus leading to selection of landmarks that may not be suitable for robot localization
- Any change in the environment causes adaptation problems due to the static nature of the approach.

In order to overcome these problems the selection of landmarks can be learned by the robot based on some simple rules.

Thrun [75] uses this approach to train a set of neural networks as feature detectors. The landmarks are chosen based on their utility for robot localization. A qualitative measure of the *goodness* of natural landmarks aims at landmarks that are stationary, reliably recognizable and sufficiently unique. Each of the neural network maps sensor input to a single value estimating the presence or absence of a particular landmark. To discover landmarks, the networks are trained so as to minimize the average a posteriori error in localization made after the robot queries its sensors. The paper presents a Bayesian probabilistic framework to measure the localization errors and estimate the robot position.

During the training phase, the robot selects a series of landmarks (that may or may not correspond to single visual features such as doors, windows and walls) and

builds a map representing the set of landmarks visible from each location in the environment. Localization is performed by using the sensory input to detect which landmarks are visible from that location and then searching in the map to see which location corresponds closest to that set of visible landmarks. The search mechanism can be a bottleneck if we want to do pure *self-localization*. In order to reduce the time required to perform localization, an incremental *position tracking* algorithm is used to update the location estimation continuously. It has the following three steps.

1. Initialization :
$$P(l) = P_i(l)$$
2. For each observed landmark vector f do :
$$P(l) = P(f|l)P(l)$$
$$P(l) = P(l)\left[\int_L P(l)dl\right]^{-1}$$
3. For each robot motion a do :
$$P(l) = \int_L P_a(l|\tilde{l})P(\tilde{l})d\tilde{l}$$

To be able to perform these operations, we only need knowledge of three probability density functions. $P_i(l)$ is the initial prior probability estimate for the robot position. $P_a(l|\tilde{l})$ describes the effects of the various robot actions and can be assumed to be known (say from expert knowledge or some learning experience). $P(f|l)$ is learned from examples in the training phase of the system. Thus the above algorithm can be efficiently used to track the position of the robot.

The major drawback of this algorithm is that it is does not solve the initial self localization problem quickly. This is because for self localization, we need to know the value of $P(l|f)$, and this is not easy to compute given the information we possess. Another potential problem is the amount of time required to train the set of feature detectors. Any change in the environment will cause us to retrain our networks, resulting in a method that is not suitable for use in dynamic environments.

Model Matching Approach

Another approach for navigation and localization in unstructured environments performs model matching between sensory inputs and an a-priori known topographical map [76]. This was the first navigational and localization system used by the Cruise Missile before the advent of military GPS [63].

Researchers at the University of Utah are at the forefront of using natural terrain features as landmarks for robot localization purposes. Thompson and Pick describe a system that detects and tracks peaks and valleys in input images to and matches these against a topographical map [76]. The strategy is to independently extract patterns from the image and the map that are likely to correspond to the same topographic features. A ridge contour finding algorithm is used to detect peaks, saddles and valleys in images. This algorithm first performs simple edge detection to extract boundary features in the image. Then a variety of thresholding and filtering algorithms are applied to this image to extract thin line segments. Peaks, saddles and valleys can then be detected in this line segment image. A template matching algorithm is then used to find precise correspondences between detected features in the image and the topographical map.

Features extracted using these processes still have a great deal of ambiguity associated with them. It is extremely hard to extract these landmarks without a-priori knowledge about the approximate viewing position and direction. Thompson cites a bottoms-up approach presented in [72] that first analyzes the uncertainty of localization at various regions in the environment, and then develops simple heuristics for selecting landmarks likely to minimize the uncertainty. These heuristics call for the robot to select landmarks that lie along a proposed navigational route rather than at random locations in the environment.

Empirical studies showed that the robot was able to use the above methodology to get a position estimate within ± 30 m of the true location. The error in the orientation was determined to be within $\pm 0.5^o$. While this method is one of the best available for navigation in unstructured known outdoor terrains, it cannot be used in unknown terrains, or even in terrains that are known but topographically *poor*. Occlusions of landmarks can also increase the uncertainty associated with the position estimate.

2.4 Multi Sensor Fusion

Strictly speaking, multisensor fusion and multisensor integration are two different processes. *Multisensor integration* refers to the synergistic use of information provided by multiple sensory devices to assist in the accomplishment of a task by a system. *Multisensor fusion* refers to any stage in the integration process where there is actual combination (fusion) of different sources of sensory information into one representational format [52]. Although the above distinction is not standard in the literature, it allows us to concentrate on a particular aspect of the general problem of combining multiple sensors (ie. sensor fusion).

Multiple sensors allow us to acquire information that is more accurate, especially if certain features are *occluded* for a particular individual sensor, in less time and cost.

These advantages correspond, respectively, to the notions of redundancy, complementarity, timeliness, and cost of the information provided the system [52].

The sensor fusion literature can be broadly categorized into two main approaches : *statistical fusion techniques* and *probabilistic fusion techniques*. Statistical fusion techniques are based on least-squares approximation methods. Some of the representative techniques in this class include least squares optimization, Kalman filtering, and regularization techniques [12], [39]. Probability theory, with its inherent notions of uncertainty and confidence, has found widespread popularity in the multisensor fusion community. The probabilistic models proposed by various researchers can be classified into four broad methodologies : Bayesian reasoning, evidence theory, robust statistics, and recursive operators [21], [39].

In the remainder of this section, we analyze some of the representative work performed using these two sensor fusion paradigms.

2.4.1 Statistical Techniques

Statistical techniques have their root in the least squares estimation. Least squares estimation is used for predicting the values of non-random variables in the system and makes no explicit assumptions about the probabilities involved in the process. It aims at finding the value of a fixed parameter x such that the the sum of the squared errors between the observed value $z(j)$ and the predicted value $h(j,k)$ is minimized. The least squares estimation rule can be expressed as :

$$\hat{x}(k) = \arg\min_{x} \sum_{j=1}^{k} [z(j) - h(j,k)]^2$$

Other estimation rules that have been proven to be equivalent to least squares estimation (under some assumptions about the underlying prior and posterior probability distribution functions) include maximum likelihood estimation, Bayesian estimation, maximum a posteriori estimation, minimum mean square error estimation and linear estimation [12].

Kalman Filter

The Kalman filter, the most popular statistical fusion operator, is a recursive linear filter and can be expressed in its simplest form as :

$$\hat{x}(k) = W'\hat{x}_{k-1} + Wz_k$$

W' and W are weighting terms that are used to remove biases and minimize error in the estimation rule. The basic version of the Kalman filter cannot deal

with non linear dynamics and measurements. This is fixed by using an extended Kalman filter that linearizes the problem around the predicted state via a second order approximation. Most Kalman filters in use are variants of the extended Kalman filter and have been modified to suit the task under study.

In [39], an extended Kalman filter for performing low level sensor data fusion is described. The paper also cites many actual implementations of sensor fusion algorithms that use this extended Kalman filter approach. Maeyama *et al.* [55] use this Kalman filter approach for performing sensor fusion for navigation in an outdoor world composed of sidewalks, trees, hedges, and walls. The recursive Kalman filter update rule is used to fuse low level position information obtained from odometry and gyroscopic data. Empirical studies show that the fused estimate of the position was always better than that given by odometry or the gyroscope alone. For more information on other applications, please refer to [39]. Here, we present the approach and framework developed for sensor fusion.

According to Kam *et al.*, the observations $z(k) \in \Re^n$ are approximated using the linear model equation

$$z(k+1) = \mathbf{H}(k+1)\mathbf{x}(k+1) + \mathbf{v}(k+1) \tag{2.1}$$

where $\mathbf{x} \in \Re^m$ is a state vector, $\mathbf{H} \in \Re^{n \times m}$ is the observation model, and $\mathbf{v} \in \Re^n$ is the observation noise in the system. The state vector satisfies a linear discrete-time state equation

$$\mathbf{x}(k+1) = \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) + \mathbf{w}(k) \tag{2.2}$$

where $\mathbf{F} \in \Re^{m \times m}$ is the system model, $\mathbf{G} \in \Re^{m \times q}$ is the sensor model, $\mathbf{u} \in \Re^q$ is the sensor input, and $\mathbf{w} \in \Re^m$ is the sensor noise. If we assume independent, zero mean, white Gaussian noise sensor processes, then equation 2.2 represents a Kalman filter recursively estimating the new state vector $\mathbf{x}(k+1)$ and equation 2.1 represents a Kalman filter predicting the next measurement given the new state vector prediction.

One can use the extended Kalman filter updating rules provided in [12] to update the state estimate $\hat{\mathbf{x}}(k+1|k+1)$ and the state covariance $\mathbf{P}(k+1|k+1)$. In lay terms, this approach can be described as an algorithm that repeatedly performs a cycle of *prediction*, *observation*, *data validation*, and *updating*.

From this discussion, it follows that Kalman filters are advantageous to use in situations when explicit accurate sensor and plant models are available and the sensor errors are negligible or well understood. They have high expressive power and implicitly encode temporal information by using past observations to predict and update estimates for the future observations. Another advantage of Kalman filters is that they readily lend themselves to decentralized architectures, with each node

responsible for prediction of local estimates. A communications protocol can be used to transfer this local information to other nodes for a global updation process [12].

The Kalman filter's main utility lies in low level fusion of *redundant* sensor data. However, there may exist situations where sensor data is not redundant but *complementary* in nature. For example, consider a decibel-meter measuring noise levels in a factory environment and a camera detecting machining equipment like high speed lathes. Humans can use observations from one modality (say hearing the lathe) to guide the search for observations in the other modality (say seeing the lathe). Kalman filters do not have an obvious extension for performing such *multi-modal* sensor fusion that is characteristic of intelligent reasoning processes.

Another drawback of Kalman filters is that they seem to fail in the presence of mobile obstacles and dynamic environments. This makes sense since Kalman filters use "weighting" measures for prediction of new states and dynamic obstacles bring in new characteristics that cannot be captured using these weighting measures [39]. Kalman filters can also not be used when sensor models are unknown and in the presence of other uncertainties. This is because they do not have the expressive power to make probabilistic predictions. Kam *et al.* advocate the use of probabilistic sensor fusion methodologies in the presence of unknown characteristics about the navigation environment or the sensor model [39].

2.4.2 Probabilistic Techniques

Bayesian reasoning and inference procedures have been widely used in other areas of sciences for a long time but have only recently gained popularity in multisensor fusion. Kortenkamp first proposed the use of a *Bayesian Network* (as opposed to statistical Bayesian methods) for performing multisensor fusion for *topological map* building in [43]. For more details on the theoretical foundations and implementations of Bayesian networks, please refer to section 3.3.1. Brooks [11] had earlier argued very persuasively for using topological maps as a means of dealing with uncertainty in mobile robot navigation. Elfes [20] later proposed another framework called *occupancy grids* which was also shown to be very efficient in dealing with the above problem. Other probabilistic approaches to sensor fusion have employed fuzzy logic and rule based systems with varied success [78].

Here, we present Kortenkamp's work in sensor fusion as leading example of Bayes Net research. We plan to use this model to develop our own framework for unified sensor fusion in the next chapter.

Bayesian Networks

Bayesian estimation has long been a popular method for sensor data fusion [39]. Bayesian networks were first used by Kortenkamp in [43] as a way of formalizing the Bayesian estimation approach. Both are based on Bayes Rule but Bayesian networks have the added feature of an explicit representation that is very powerful and captures causal relationships between various entities in the system.

A simple sensor fusion Bayesian network can be easily designed by representing the sensor data from the individual sensors as the *inputs* and a unified map representation as the single *output*. The degree of confidence in each sensor's measurements can be represented using the conditional probability matrices associated with the causal relationship linking the sensors to the unified map. These conditional probabilities do not have to be known a-priori and can be *learned* using statistical sampling techniques or supervised learning approaches.

One of the most popular approaches to sampling techniques uses Gibbs sampling for generation of training data [64]. In this approach, a small set of complete training data (sensor observations and actual environment state) serves as a base for a sample generator to generate a large set of training data from incomplete samples. The incomplete samples can have missing sensor observations or missing environment observations. Supervised learning is done by placing the mobile robot at various locations in the environment and polling the sensors. The sensor observations and actual observations are compiled in contingency tables. These contingency tables encode empirical expert knowledge about the reliability of each sensor in correctly observing the environment. When a sufficiently large amount of sample data is gathered, this conditional probability table can be calculated from the contingency tables.

The most important (and one of the first) work using Bayesian networks for sensor fusion was first published by Kortenkamp in [43]. Kortenkamp uses Bayesian networks for combining sensor data from sonar and vision sensors to build a topological map of the world. Topological maps were chosen because of the ease of expressing uncertainties in their framework and because of their compactness. Topological maps also do not have the notion of a global coordinate system and all movements occur locally.

The features selected for representation in the topological maps are *gateways* that can be easily recognized by the vision system and the sonar system. Since the major focus of the work was on sensor fusion, these gateways were very simple features characterized by sharp vertical edges making them easy to recognize. Figure 2.2 shows some examples of gateways used in the system.

The sonar and vision sensors are fine-tuned to recognize these gateways easily and build a topological map of the world by storing information about these transition areas. Some properties of gateways are :

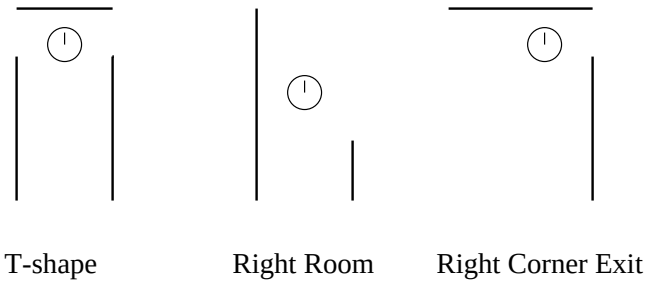

Figure 2.2: A few examples of different types of gateways.

1. they tend to be places that are visited frequently
2. they open up new views for the robot sensor systems
3. they have a limited number of directions for robot movement

Gateways have been proven to be an efficient landmark for indoor navigation because of the existence of such structures in man-made environments. The first step in the sensor fusion algorithm recognizes gateways and classifies them based on their shape. Kortenkamp identified twenty five locations in the test environment that were detected by the vision and sonar sensors as gateways. The need for vision sensors arises because sonar sensors cannot differentiate between two gateways with the same shape. Thus a visual representation of the scene at each gateway is also stored in the topological map.

Now, the robot localization problem can be solved by combining a scene (from the vision sensor) and a gateway (detected by the sonar sensor) at every place in the topological map. This combination is done using a very simple Bayesian network shown in figure 2.3.

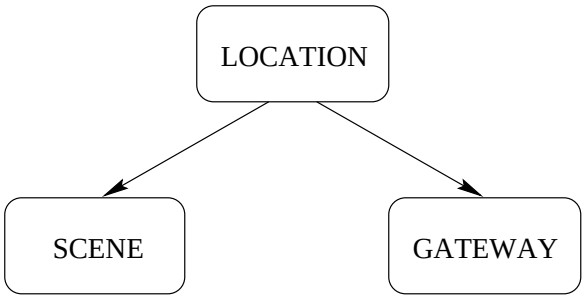

Figure 2.3: The Bayesian network used for place recognition.

	Loc A	Loc B	Loc C	Loc D	Loc E	Loc F	Loc G
Sce A	**.43**	.09	.22	.05	.05	.10	.06
Sce B	.05	**.52**	.21	.06	.05	.05	.05
Sce C	.10	.12	**.36**	.20	.04	.13	.04
Sce D	.14	.05	.24	**.43**	.05	.04	.05
Sce E	.14	.14	.14	.14	.14	.14	.14
Sce F	.14	.14	.14	**.16**	.14	.14	.14
Sce G	.14	.14	.14	.14	.14	.14	.14

Table 2.1: Likelihoods for each place using only vision.

	Loc A	Loc B	Loc C	Loc D	Loc E	Loc F	Loc G
Son A	**.82**	.04	.04	.04	.04	.04	.04
Son B	.02	.31	.31	.31	.06	.00	.00
Son C	.02	.31	.31	.31	.06	.00	.00
Son D	.02	.31	.31	.31	.06	.00	.00
Son E	.04	.12	.12	.12	**.61**	.00	.00
Son F	.00	.00	.00	.00	.00	**.90**	.10
Son G	.00	.00	.00	.00	.00	.10	**.90**

Table 2.2: Likelihoods for each place using only sonar.

In the experiments conducted by Kortenkamp, the robot identified seven unique locations and associated a scene and a gateway with each location. In the Bayesian network shown in figure 2.3, this was represented as two conditional probability matrices describing the likelihood of the system being at a certain location given the evidence from a scene and a sonar measurement respectively.

Tables 2.1 and 2.2 give the individual conditional probabilities that were observed by the robot in [43]. Neither the vision, nor the sonar sensor can independently recognize every location in the world unambiguously given their sensor measurements. However, using the Bayesian network of figure 2.3 results in the combined conditional probability table shown in table 2.3. Using this combined likelihood table, we can uniquely determine the robot's location given the scene and the gateway that the sensors are observing. The conditional probabilities in the tables are the normalized match percentages. For example, if there are three scenes and the match percentages are 0.25, 0.90 and 0.75 then the conditional probabilities would be 0.13, 0.47, 0.40. The tables were not *learned* using sampling or supervised learning techniques but determined using mostly expert knowledge (prone to errors resulting from assumptions

	Loc A	Loc B	Loc C	Loc D	Loc E	Loc F	Loc G
Sen A	**.95**	.01	.02	.01	.01	.00	.00
Sen B	.00	**.65**	.26	.07	.01	.00	.00
Sen C	.00	.17	**.52**	.29	.01	.00	.00
Sen D	.01	.07	.33	**.58**	.01	.00	.00
Sen E	.04	.12	.12	.12	**.61**	.00	.00
Sen F	.00	.00	.00	.00	.00	**.90**	.10
Sen G	.00	.00	.00	.00	.00	.09	**.91**

Table 2.3: Combined (sonar and vision) likelihoods for each place.

and incorrect knowledge).

The advantages of the above system are trifold. We achieve higher accuracy in place recognition, more robust performance in terms of sensor errors and the ability to disambiguate between different locations. However, the above method also has a lot of limitations. Firstly, the above system works only in highly orthogonal, well structured, indoor environments. The real world is typically unstructured and does not have the nice orthogonal features that can be detected by sonar sensors. Also, outdoor navigation presents an entirely different problem arising from other sensor modalities such as inclination and declination, mobility of other objects in the world, lack of good visual cues for scene interpretation and the like. To solve autonomous navigation problems in such situations would require the use of many more sensors besides vision and sonar.

Secondly, the experimental space of [43] consisted on only seven distinct gateways. There is a question of how well the system would scale to larger spaces. It is certainly reasonable to expect system degradation as the space size increases. It is not quite clear how a Bayesian network could adapt to increasing space sizes. A simple Bayesian network such as the one described in figure 2.3 would certainly not suffice to capture the ambiguities of a complex system. We propose a modified approach to sensor fusion using Bayesian networks in the next chapter and characterize its properties in detail. The modifications are aimed at removing the restrictions on the above approach.

2.5 Motion Planning in Dynamic Environments

Motion planning refers to the computational process associated with moving a mobile robot from one location in the environment to another. It is necessary to differentiate between motion planning and path planning at this stage. *Path planning* is a motion

planning technique where a-priori information about the environment is used to explicitly generate a path that moves the robot from an initial to a goal location. The other motion planning technique is reactive navigation, where such a global path is not generated. Motion planning is accomplished via selecting from a series of actions whose global effect is the movement of the robot from the initial location to the final location [27]. The capability of motion planning in dynamic environments is very important for autonomous robots.

A relevant issue that plays an important role in deciding which motion planning strategy to employ is *information about the environment*. When all information regarding obstacle sizes, locations, motions etc. is known a-priori, we can use a path planning approach to generate a collision free path to reach the goal location [7]. However, when there is unpredictability in the estimation of these attributes of an obstacle, a path planning approach is not guaranteed to generate a collision free path [1]. In this case, a reactive or *hybrid* approach to motion planning is employed.

2.5.1 Reactive Approaches

Reactive approaches to motion planning were the first ones to be used in dynamic environments. A major reason for this is that path planning in the presence of dynamic obstacles is a very hard problem. So in keeping with chronological history, we discuss these approaches first. Most reactive approaches to motion planning are souped up collision detection and avoidance methodologies [50].

The most common collision detection and avoidance strategies work by creating a *personal space* around the robot and detecting obstacles that cross into this space. Once an obstacle is detected, and obstacle avoidance algorithm is executed that steers the robot away from a possible collision. There are many different approaches to performing collision avoidance. Some of the simplest approaches utilizes a set of rules that define the robot's actions given the obstacle's location and path of motion [27]. For example, if a robot detects an obstacle to its left, and there is no obstacle on its right, then the collision avoidance rule can guide the robot to move to the right to avoid the obstacle. Such rule based collision avoidance approaches lend themselves quite easily to fuzzy logic reasoning systems for collision avoidance [50]. While such systems are easy to use, they are built using purely ad-hoc techniques that can be very environment dependent. We have implemented a similar obstacle avoidance system on the mobile wheelchair platform. This work is described in more detail in section 4.1.

Fuzzy Logic Systems

Fuzzy logic control is based on the theory of fuzzy sets [50]. A fuzzy set A in a universe of discourse X is defined by its membership function $\mu_A(x)$. For each $x \in X$, there exists a value $\mu_A(x) \in [0, 1]$ of the membership function representing the degree of the membership of x in X. In fuzzy systems, membership functions associated with linguistic variables are used to fuzzify physical quantities. A fuzzy rule base that characterizes the relationships between fuzzy inputs and fuzzy outputs is created. A simple fuzzy rule relating a fuzzy input a to a fuzzy output b can be specified as:

$$\textit{If } a \textit{ is } W \textit{ Then } b \textit{ is } Y$$

where W and Y are the fuzzy values associated with fuzzy variables a and b. The response of each fuzzy rule is then weighted according to the degree of membership of its fuzzy inputs. The inference engine provides a set of fuzzy control actions according to these weighted responses (a classification of controls based on inputs and outputs with interpolation in between deterministic mappings). A defuzzification technique is then used to extract a deterministic control action from the fuzzy logic controller. A commonly used technique extracts a control action based on the centroid values of the membership functions of each fuzzy variable as follows:

$$\bar{u} = \frac{\sum_{i=1}^{n} \mu_Y(c_i) * c_i}{\sum_{i=1}^{n} \mu_Y(c_i)}$$

where $\bar{u}$ is a deterministic output value of the fuzzy controller, n is the number of fuzzy control rules, and c_i is the centroid of the membership function associated with each linguistic value in the output space.

Learning Fuzzy Control

Fuzzy logic systems seem to work well for obstacle avoidance tasks in domains for which they were designed. Since the design process is usually ad-hoc, it is very hard to maintain a fuzzy control system for obstacle avoidance if the environmental constraints change. To alleviate this problem, Aoki *et al.* proposed a reinforcement learning approach that generates an optimal set of fuzzy rules [1]. The initial training phase allows the robot to *explore* its environment and learn how to avoid obstacles by providing a *reward or punishment* feedback when it avoids or collides with obstacles respectively. The only information coded a-priori into the control is the set of actions that can be performed by the robot. The fuzzy rules are learned entirely during the learning phase.

Such a fuzzy rule selection scheme allows a robot to learn a fuzzy control algorithm that is optimal in terms of avoiding collisions. The system was tested on

an autonomously moving Khepera robot in the presence of three randomly moving obstacles (also Khepera robots but moving randomly with no collision detection or avoidance). The sensors are polled every 0.2 seconds to update the robots view of its surrounding environment. Empirical studies show the effectiveness of this approach with completely collision free trajectories for the robot in the presence of moving robots.

Beom and Cho [7] had earlier demonstrated how to use reinforcement learning techniques to learn obstacle avoidance and goal seeking behaviors. Their approach learned deterministic rules for the two behaviors and then used fuzzy logic to determine which behavior to execute at a given time.

While the above approaches alleviate the problem of maintaining a robust fuzzy rule based obstacle avoidance system, the system still needs to be retrained each time the environment changes. Another drawback of Aoki's approach is that it may take a long time for a robot to reach its goal as the shortest collision free path may not be followed. While Beom's approach does not suffer much from that limitation (because of the goal seeking behavior), it does not have the nice properties of fuzzy obstacle avoidance. Also, all of the reactive approaches tend to quickly fail in the presence of U shaped obstacles which can cause the robot to reach a deadlock case or go around in cycles.

2.5.2 Planning Approaches

Path planning approaches aim to generate a collision free (set of) path(s) that the robot can use to move from an initial location to a final location. The planning strategy can be global (generates a complete path from the initial to the final location) or local (generates smaller paths that take the robot closer to the goal). We have already noted that the problem of path planning in the presence of dynamic obstacles is NP-hard, even when the obstacles are convex polygons moving with bounded constant linear velocities without rotation [13], [67]. However, this has not stopped researchers from proposing many different path planning algorithms made tractable by imposing certain additional constraints on the system.

The Basic Path Planner

The canonical path planning task in a static a-priori known environment can be performed in three steps [42] :

1. *Free Space Generation* : The free space regions of the environment, taking into account the positions of the robot and the obstacles is generated.

2. *Model Building* : A representation such as a visibility graph which models the free space regions is created.

3. *Solution Path Searching* : A search method finds the shortest collision free path from the model or representation constructed in the previous step.

Ko *et al.* give an example of a path planner based on triangulation of obstacles for model generation [42]. The planner first reduces the robot to a point object by correspondingly enlarging the obstacles (the configuration space method) and generates a graph that models the free space as triangles with obstacles at each of the vertices (the incenter of each triangle forms a node in the model graph). A fast graph search mechanism is then utilized to extract the shortest path from this representation.

The first attempts at extending the path planning approach to dynamic environments using fuzzy logic systems that switched between a path planning behavior and an obstacle avoidance behavior. The path planner computes a path from the current location to the goal location and the fuzzy control navigates along this path until an obstacle is detected. The system then switches over to the obstacle avoidance behavior. Once the robot's proximity is determined to be clear of obstacles, the path planner is evoked again to generate a new path from the robot's current location to the goal [27]. Obviously, this approach is very inefficient when many mobile objects are present in the environment since the path planning algorithm is restarted every time a moving obstacle is detected and avoided.

Time Minimal Path Planning

To incorporate the characteristics of dynamic obstacles into the path planner, *time enlarged* approaches to the planning problem were proposed [27]. To keep the space and computation tractable, most of these methods perform local path planning instead of global path planning.

Fujimura describes such a time-minimal path planner in [27]. The first step involves construction of an *accessibility graph* that captures the movements of all the entities in the system. The graph consists of annotated nodes, each node representing an object (the robot, the obstacles or the goal) in the environment. The node is annotated with information regarding the motion characteristics of each object. During execution, the system constructs an online graph (actually a tree rooted at the robot) that contains all the objects *visible* to the robot and considered *close enough* to represent danger to it. A path planner that looks at this graph and computes a time-minimal (longest distance in shortest time) path given the current accessibility graph. A sequence of these time-minimal paths moves the robot from the initial to a goal position. Empirical studies have shown this system to be very robust even in the presence of a large number of moving obstacles. The system is able to perform

path planning even when the motion characteristics of objects are non-linear. The major drawbacks of this approach are that the motion of all the objects has to be completely known a-priori, and the amount of computational effort required to generate a path quickly increases as more obstacles with complex motions are added to the environment.

Tsubouchi[77] uses an approach very similar to that proposed by Fujimura for performing path planning when the motion of the obstacles is partially known. A motion planning algorithm repeats the following steps in a continuous cycle :

1. Observation : Observe moving obstacle motions and compute their position and velocity.
2. Forecast simulation : Forecast obstacle motions using observed position and velocity and motion model.
3. Path planning : Select feasible path avoiding collision in the near future.
4. Navigation : Send commands to navigation controller to move robot along selected path.

The motion model mentioned in step 2 is constructed from available knowledge about the motion characteristics of the obstacles. Having a better knowledge of the obstacle motions produces a better motion model which in turn provides the motion planner with a more accurate forecast of the obstacles current motion. To relax the assumption on prior knowledge, the system is equipped with a learning algorithm that updates the motion model of the obstacles by observing their positions and velocities over time. This learning algorithm runs in parallel to the motion planner.

For the motion planning algorithm to be feasible, it must be able to execute a complete cycle in a short time interval. To accomplish this, Tsubouchi restricts the domain via a number of assumptions and constraints. These include

- The robot and the obstacles are all circular in shape and can be detected easily.
- The robot knows it position accurately and can compute the position and current velocity of the obstacles from sensor data quickly.
- There is no constraint on the (time or length) cost of path taken from the initial to the goal location.

A simulator developed under these assumptions repeats the motion planning cycle every 0.2 seconds. Simulation results show that the robot is able to perform good collision free motion planning in the presence of unknown dynamic obstacles. The learning algorithm takes about 30 trials to produce a limited motion model to aid in prediction of obstacle velocities.

Statistical Path Planning

Some recent work on path planning in the presence of unknown or partially known obstacles has been performed by Gutsche [31]. This approach is similar to that proposed by Fujimura, except that a global planner is also incorporated into the system.

This global planner generates an initial path taking into account all the stationary objects in the system and the dynamic objects with partially or completely known motion characteristics. The global path is generated only once (during the initialization phase) so that the computation time required for the path planner is not a bottleneck when the robot is performing navigation. Once an initial path has been generated, the control system attempts to perform navigation along this route by making local modifications to the path at run time. Using a global planner that does not take dynamic objects into account cannot produce a good collision free path (even with local modifications).

Gutsche uses *implicit rules* that capture statistical information representing average behavior of dynamic objects to overcome the above problem. In this case, the statistical information includes precomputed occupancy probability and statistical motion flow for all the dynamic objects in a grid based representation of the world. Intuitively, one can think of implicit rules as representing knowledge deemed to be the normal behavior, eg."cars driving on the same side of the road are moving in the same direction". While implicit rules do not have to be true always, they are true for the average case (or a majority of time). Also, these implicit rules can change in local behavior, eg. the above mentioned implicit rule does not hold statistically true on a single-lane two-way road. Using a grid based approach, different sets of implicit rules can be associated with different locales in the environment.

Simulation results show this approach to be very efficient, even in the presence of a large number of completely unknown obstacles. The paths followed by the robot to move from the initial to the goal state are very close to the optimal path that could have been followed if all the dynamic obstacles were completely known and a path planner was used to generate the path. The drawback of this approach is that the performance quickly degrades in the presence of obstacles that do not exhibit behavior in compliance with the implicit rules. It may also be very hard to produce a good set of implicit rules for a sufficiently unconstrained or general environment.

Chapter 3

A Generalized Framework for Robot Navigation

3.1 Introduction

Before we present our approach to a generalized framework for autonomous mobile robot navigation, we would like to motivate some of the reasons for performing research in this area. An extensive survey of the vast body of literature on autonomous mobile robot navigation leads us to conclude that:

1. Most of the research effort is very focused in its scope and attempts to create solutions for specific environments characterized by their unique constraints and not readily generalizable.

2. There exists a vast body of literature specifying multiple techniques for solving problems such as robot localization, goal specification, goal recognition, motion planning, multi sensor fusion, and dynamic obstacle avoidance in different environments with different constraints. While the robustness and efficiency of many of these techniques is very questionable, they nevertheless give us a foothold into building navigation systems capable of operating in any environment.

3. There exist no autonomous mobile robot navigation systems that are completely generalized and exhibit robust performance under different sets of environmental and robot constraints and assumptions (we define this in greater detail in the next section).

We believe that while a single technique cannot be used to create a robust and efficient generalized autonomous mobile robot navigation system, a combination of

some of the different methods in a probabilistic framework can achieve this goal. One question that needs to be answered at this stage is "What is a generalized navigation system?"

Generalized System Defined

We define a *generalized autonomous mobile robot navigation system* as a system that is capable of performing all the various robot navigation related tasks in a robust and efficient manner, without placing any constraints on the characteristics of the environment or the robot. These robot navigation tasks are described in detail earlier in this report and include robot localization, goal specification and recognition, motion planning, obstacle avoidance, and sensor fusion.

The operating environment of a generalized mobile robot navigation system can include:

1. outdoor and indoor locations
2. rough and smooth terrain
3. stationary and moving obstacles
4. structured and unstructured landmarks
5. single or multiple robots
6. varying sensors and sensor models

No assumptions are made on the a-priori knowledge about any of the features mentioned above. A generalized system must be able to perform navigation when the environment and all its characteristics are completely known (either a-priori or through learning) and when they are completely unknown.

It is easy for the reader to grasp the hardness of creating such a generalized robot navigation system. We do not propose to create such a system, but instead propose a framework that can be used for creating a navigation system for any environment specified. While this navigation system may still be environment dependent, it is hoped that the framework will provide the designers with an easy and efficient methodology for adapting the navigation system to changes in the environment.

Generalized Framework Defined

We define a *generalized framework* as a methodology that allows easy adaptation of a mobile robot navigation system to changes in the constraints and assumptions

associated with the operating environment. The primary task of a generalized framework is to create a navigation system that can perform robot localization (absolute or relative), multisensor fusion, and motion planning given the characteristics specifying the environment. This environment can possess any subset of the characteristics associated with a generalized mobile robot navigation system defined in section 3.1.

A generalized framework provides a representation for storing information acquired about the system, either through prior knowledge or while performing navigation. It also provides a set of *tools* for updating the information stored in this representation. Along with the representation and the tools, there also exists a methodology that defines how the tools can be used to update the representation.

The Proposed Framework

We propose a probabilistic framework that is based on the generalized framework approach presented in section 3.1. Probabilistic systems capture the uncertainties present in the environment very well, and hence make good candidates for a generalized framework. These uncertainties can arise due to lack of a-priori knowledge abut the environment (ie. no map or inaccuracies in map), presence of dynamic obstacles, and limitations of the sensor systems among other reasons.

The representation chosen for modeling the environment is a modification of the *Occupancy Grids* proposed by Elfes in [20]. Though occupancy grids are a recent contribution to the representation literature, they have gained instant popularity among researchers developing probabilistic navigation systems [6], [21]. We use a modified version of the occupancy grids, called *dynamic occupancy grids* as our representation of the environment. Section 3.2 presents the details of this representation.

Pearl describes a Bayes rule based network approach for modeling causal relationships in [64]. This approach, called Bayesian Networks, explicitly encodes uncertainty information associated with related entities in a system and describes the interactions between these entities in a very clear and formal manner. We use a modification of these Bayesian networks, termed *dynamic Bayesian networks* as our tool for updating the dynamic occupancy grid representation. Details of dynamic Bayesian networks are presented in section 3.3.

The last component of our proposed framework is the methodology for updating the dynamic occupancy grid using dynamic Bayesian networks. To relax synchronization issues and constraints, we use an asynchronous update policy that uses dynamic Bayesian networks to create new probability density functions (PDF) that are then used to update the information stored in the occupancy grids. Whenever a robot needs to perform some new motion task, it discretizes the PDFs currently associated with the dynamic occupancy grids to create a unified *view* of the current state of the environment. A suitable motion planning approach can then be used to perform

robot navigation.

3.2 Dynamic Occupancy Grids

A representation used for modeling the physical attributes of the environment and the entities present in it must possess certain qualities. It must be powerful enough to express all the entities that need to represented. At the same time, it must be adaptive in that only entities that effect the navigation algorithm at any given stage need to be represented during that stage. Thus, if there is a road present in the environment, but the robot is nowhere near it, we do not need to store information about it in the representation. Such an approach restricts the representation space from growing exponentially, allowing for implementation of tractable algorithms for motion planning.

Another desired characteristic of any representation is that it must be able to quickly update its knowledge about the current state of the environment without too much computational effort. This is especially important in our approach since we rely purely on the information present in the dynamic occupancy grids for performing navigation.

Dynamic Occupancy Grids (DOG) are based on the occupancy grid framework proposed by Elfes in [20]. A description of the occupancy grids, presented in [20], follows.

3.2.1 Introduction to Occupancy Grids

Occupancy grids are a stochastic tessellated representation of perceived spatial information about the robot's operating environment. The environment is explicitly divided into small square spatial areas called *cells*. Each cell stores a probabilistic estimate of the occupancy of that cell in the form of a *state variable*. The representation also implicitly captures spatial adjacency information.

In a single dimension occupancy grid, each binary state variable, $s(C_i)$, predicts the presence or absence of an object in a cell C_i. Elfes uses the labels OCC and EMP to identify the two possible states for each cell, *occupied* and *empty*. Since the states are exclusive and exhaustive, $P[s(C_i) = OCC] + P[s(C_i) = EMP] = 1$. Bayesian estimation rules are used to update the values associated with the state variables.

First, a sensor model of the form $P[r|z]$ is used to interpret a sensor measurement r with reference to the actual measurement z. A new estimation of the occupancy of a cell C_i is given by the value of $P[s(C_i) = OCC|r]$. Using Bayesian estimation rule, this condition probability can be computed as follows :

$$P[s(C_i) = OCC|r] = \frac{P[r|s(C_i) = OCC].P[s(C_i) = OCC]}{\sum_{s(C_i)} P[r|s(C_i)].P[s(C_i)]} \tag{3.1}$$

The denominator term $P[r|s(C_i)]$ in equation 3.1 can expressed using the sensor model as:

$$P[r|z] = P[r|s(C_i) = OCC \wedge s(C_k) = EMP, k < i] \tag{3.2}$$

The value of $P[r|s(C_i)]$ terms can be computed from a-priori information about the environment using equation 3.2 as follows. We first calculate the conditional distribution of r for each possible state (OCC and EMP) using:

$$\begin{aligned} P[r|s(C_i) = OCC] \quad &= \sum_{\{G_{s(C_i)}\}} \quad \Big(P[r|s(C_i) = OCC, G_{s(C_i)}] \\ &\qquad \times P[G_{s(C_i)}|s(C_i) = OCC]\Big) \end{aligned} \tag{3.3}$$

where $\{G_{s(C_i)}\}$ represents all possible grid configurations with $s(C_i) = OCC$. $P[r|s(C_i) = EMP]$ is obtained using a similar equation. The configuration probabilities, $P[G_{s(C_i)}|s(C_i)]$ are obtained from a-priori knowledge. This a-priori knowledge can be learned experimentally or simply *ignored* by setting assigning equal prior probabilities to the occupancy condition of each cell.

$$P[s(C_i) = OCC] = P[s(C_i) = EMP] = 1/2 \tag{3.4}$$

Using the cell independence assumption, these priors can be easily used to determine the configuration probabilities. The sensor model is then used to generate the conditional distribution of r using the equation 3.2. Intuitively, we are obtaining the conditional distribution of r by setting the state of a particular cell C_i, varying the states of the other cells in the occupancy grid, and applying the sensor model to each configuration. The above approach to updating one dimensional occupancy grids easily extends to higher dimensional grids.

The major advantage of this approach is that the navigation system only needs to look at the occupancy grid to see what cells around the robot's location are probabilistically free of obstacles to move the robot towards a coordinate specified goal. If all the sensor models, and the prior probability distributions of the configuration space are learned or known a-priori, then the method performs very fast updates of the occupancy grid using the sensory data.

However, the above approach also has some drawbacks. Firstly, sensor models need to be known accurately to do updating of the occupancy estimates in the occupancy grids. This is usually not possible due to two reasons : (1) weak, ill-designed sensors and (2) multiple dependent sensors. Multiple independent sensors whose

individual sensor models are known can be handled at some computation cost by serializing the updating operation for all the sensors. The second drawback is that there exist subsets of configurations that are indistinguishable under a single sensor observation. In this case, numerical solutions to the above equations may not computable, and the navigation system may have to rely on close-formed solutions [20].

3.2.2 Modified Occupancy Grids

We use the occupancy grid approach developed by Elfes [20] and modify it to represent a more generalized environment than the one represented by Elfes as follows:

1. We associate a set of *state vectors* with each cell (instead of a single state variable). There is a state vector associated with each entity currently represented by the occupancy grids. Each state vector contains probabilistic estimates of an object's identity, location and characteristics (such as velocity, acceleration, or other behavior). Thus, each cell of our modified occupancy grid captures a lot more information about the environment.

2. We associate a set of global probability distribution functions (PDF) with the occupancy grid. These PDFs encode the value of the probabilistic measures represented in the state vectors. There is a set of PDFs for each entity in the system, encoding the current probabilistic estimates for the entity's identity, location and characteristics.

3. We make our modified occupancy grids *dynamic* by allowing the set of represented entities to change over time. Only those entities currently affecting the navigation system are included in the representation. The set of entities represented at any given time is obtained by using an independent process that reads the sensors and constructs a set of detected entities. This provides us with major configuration space reduction as the navigation system is only concerned about events occurring in some pre-defined neighborhood around the robot. This adaptive representation seems especially well suited for entities that only have a limited presence in the environment (eg. cars driving in a direction orthogonal to the robot).

4. The discretization step that converts PDFs to state vectors is performed when the navigation system looks in the dynamic occupancy grid to determine the current state of the environment. This approach saves us time as PDFs are only discretized when they are used and not when they are updated. Also, this discretization occurs only for cells in the robot's neighborhood, resulting in more time savings.

5. A decay function is associated with the PDFs that *flattens* them when they have not been updated for a period of time. Figure 3.1 and 3.2 show two PDFs *overlayed* on an occupancy grid. The fatter PDF represents a PDF that has not been updated for a while. The thinner PDF represents a recently updated probability estimate.

6. We change the updating procedure by using these PDFs and discretizing their values in each cell to obtain the information for the state vectors directly. These PDFs are generated using dynamic Bayesian networks described in section 3.3 instead of a Bayesian estimation rule. The advantages of using dynamic Bayesian networks are analyzed in the same section.

Next we describe the dynamic Bayesian networks used to update the PDFs associated with the dynamic occupancy grids.

3.3 Dynamic Bayesian Networks

We need a method to generate the PDFs used in the dynamic occupancy grids for estimation of entity characteristics. Such a method must possess certain qualities to be applicable to any general environment. Firstly, it must explicitly deal with uncertainties as the outputs of this method are PDFs. Secondly, it must be easy to construct or build the method for producing these PDFs based on the sensory observations (our only source of information about the environment). Thirdly, the time required to construct a new set of PDFs based on sensor data received must be minimal to allow the dynamic occupancy grids to quickly reflect changes in the environment. Lastly, since our dynamic occupancy grids can represent different entities (and this requires different PDFs) at different times, the method used to generate these PDFs must also be adaptive in the same sense.

Pearl describes Bayesian networks in [64] and motivates their use for tasks that require causal relationships to be modeled. The PDFs required by the dynamic occupancy grids are definitely causally dependent on the sensory inputs. Hence, we choose to explore and develop a Bayesian network based tool for updating the PDFs used by the dynamic occupancy grids.

3.3.1 Introduction to Bayesian Networks

Bayesian networks are directed graphs that represent the *conditional independencies* and *causal relationships* between various entities in the system. The links between the entities represent the conditional probabilities (or likelihoods) of inferencing the existence of one entity (the destination of the link) given the existence of the other

entity (the source of the link). Each entity can have many such directed inputs and outputs, each specifying its dependence relationship to the entities the inputs originate from and the output go to.

Given this definition, a Bayesian network can be thought of as a knowledge base. It explicitly represents our *beliefs* about the system and the relationships between the various entities of the system. Pearl presents a detailed description of Bayesian networks and associated theory in [64] which we summarize here.

Bayesian networks operate by propagating beliefs through the network once some evidence about the existence of an entity is asserted. When we assert the existence of an entity, we can propagate this belief *upwards* and *downwards* in the network by calculating posterior probabilities of the existence of all other entities connected to the asserted entity. Consider the Bayesian network described in Figure 3.3.

Various dependencies in the system are depicted via the directed links. Our belief in the existence of earth depends upon the existence of land and water. Further, our belief in the existence of land depends upon the existence of rock and sand. These beliefs have associated with them, condition probability matrices that represent expert knowledge about the dependencies between the two entities on the link. For example, the conditional probability matrix on the link from Land to Rock represents the following expert knowledge:

- $P[Rock|Land] = 0.6$
- $P[\neg Rock|Land] = 0.4$
- $P[Rock|\neg Land] = 0.2$
- $P[\neg Rock|\neg Land] = 0.8$

The conditional probabilities reflect our knowledge of the system and can be derived using sampling or learning methods. This is presented in section 2.4.2. The same section also contains an implementation of Bayesian networks for simple sensor fusion of sonar and vision sensory information.

3.3.2 Belief Propagation in Bayesian Networks

Bayesian networks work by propagating evidence received in *support* of some node in the network to all the other nodes in the network. At the end of this belief propagation, the new beliefs about the various entities represented by the nodes in the network are generated. For our analysis here, we deal with discretized probabilities. However, Pearl shows how the very same technique can also be extended to propagate continuous PDFs rather than discretized probability arrays [64].

Consider a general Bayesian network shown in figure 3.4. This Bayesian network shows a typical node $\mathbf{X}$ with one parent, $\mathbf{U}$ and m children, $\mathbf{Y}_1$, $\mathbf{Y}_2$, ..., $\mathbf{Y}_\mathrm{m}$. The following three measurements are necessary and sufficient to compute the *Belief* of node $\mathbf{X}$.

1. The current *causal* support, $\pi_X(u)$, contributed by the parent, of $\mathbf{X}$.

$$\pi_X(u) = P(u|e_X^+) \tag{3.5}$$

2. The current *diagnostic support*, $\lambda_{Y_j}(x)$, contributed by the j^{th} child, of $\mathbf{X}$.

$$\lambda_{Y_j}(x) = P(e_{Y_j}^-|x) \tag{3.6}$$

3. The fixed conditional probability matrix $\mathbf{M}_{x|u} = P(x|u)$ that relates the variable $\mathbf{X}$ to its immediate parent $\mathbf{U}$.

We now define the Belief of a variable x as

$$\mathrm{BEL}(x) = P(x|e_X^+, e_X^-) \tag{3.7}$$

Using Bayes rule to expand the right hand side of equation 3.7, we get

$$\mathrm{BEL}(x) = \alpha P(e_X^-|x)P(x|e_X^+) \tag{3.8}$$

where $\alpha = [P(e_X^-|e_X^+)]^{-1}$ is a normalizing constant. Substituting equations 3.5 and 3.6 in equation 3.8, we get

$$\mathrm{BEL}(x) = \alpha\lambda(x)\pi(x) \tag{3.9}$$

Now, to compute $\mathrm{BEL}(x)$, we need to be able to compute the values of $\pi(X)$ and $\lambda(X)$. These can be computed as follows ...
To compute $\pi(x)$, we use equation 3.5 and get

$$\begin{aligned} \pi(x) &= P(x|e_X^+) \\ &= \sum_u P(x|e_X^+, u)P(u|e_X^+) \\ &= \sum_u P(x|u)P(u|e_X^+) \\ &= \mathbf{M}_{x|u}\pi_X(u) \end{aligned} \tag{3.10}$$

To compute $\lambda(x)$, we use the conditional independence assumption that the children of $\mathbf{X}$ are conditional independent of each other unless $\mathbf{X}$ is instantiated. Then we can assume that $e_X^- = e_{Y_1}^- \cup e_{Y_2}^- \cup ... \cup e_{Y_m}^-$. Using equation 3.6, we get

$$\begin{aligned} \lambda(x) &= P(e_X^-|x) \\ &= P(e_{Y_1}^-, e_{Y_2}^-, ..., e_{Y_m}^-|x) \\ &= P(e_{Y_1}^-|x)P(e_{Y_2}^-|x)...P(e_{Y_m}^-|x) \\ &= \lambda_{Y_1}(x)\lambda_{Y_2}(x)...\lambda_{Y_m}(x) \end{aligned} \tag{3.11}$$

Note that this product rule also applies when $\mathbf{X}$ is instantiated but the proof is much more involved. Now we know how to compute the values of $\pi(x)$ and $\lambda(x)$ given the conditional probability matrix and the *messages* received from the parent and the children. The belief propagation algorithm consists of three rules that are executed asynchronously:

1. Belief Updating : node $\mathbf{X}$ updates its Belief via the following rule

$$\text{BEL}(x) = \alpha\lambda(x)\pi(x)$$

2. Top-down Propagation : node $\mathbf{X}$ computes new π messages to be sent to its children via the following rule

$$\pi_{Y_j}(x) = \alpha\pi(x)\prod_{k\neq j}\lambda_{Y_k}(x)$$

3. Bottom-up Propagation : using the λ_{Y_i} messages received from its children, node $\mathbf{X}$ sends its parent $\mathbf{U}$ a new message $\lambda_X(u)$ computed via the following rule

$$\lambda_X(u) - \sum_X \lambda(x)\mathbf{M}_{x|u}$$

There arise four special cases that are not handled by the above belief propagation algorithm for node $\mathbf{X}$. These occur at *leaf* and *root* nodes and when a particular node is instantiated.

1. Anticipatory node : A leaf node that has not been instantiated. Its Belief should be equal to π so set λ = (1,1,...,1).

2. Evidence node : A node that has been instantiated. If the j^{th} label of the node is observed to be true (with probability = 1), we set λ = (0,...,0,1,0,...,0) with 1 at the j^{th} position.

3. Dummy node : A node **Y** representing judgemental evidence bearing on **X**. We do not specify $\lambda(y)$ or $\pi(y)$ message but send a message $\lambda_Y(x)$ to **X** where $\lambda_Y(x) = \beta P(evidence|x)$, β being a convenient normalization constant.

4. Root node : The boundary condition for root node is established by setting π(root) = prior probability of the root variable.

The above belief propagation approach works with minor modifications when the network is shaped like a poly-tree or has undirected cycles in it. We plan to use a continuous version of the PDFs for propagation but restrict the networks to singly rooted trees as they are easy to work with.

3.3.3 Modified Bayesian Networks

We propose the following extensions to the Bayesian network approach described in the previous section:

1. Addition of a history node as shown in figure 3.5 linked to the corresponding node in the Bayesian network to explicitly encode a temporal aspect into the Bayesian network.

2. Dynamic structure changes in the Bayesian network to represent human-like reasoning strategies. These structural changes are triggered by the Beliefs crossing a threshold value. One such example is shown in figure 3.6.

3. Run time selection of Bayesian networks (from a pre-developed or learned library of networks) to be used at any given time based on the entities represented in the dynamic occupancy grids at that same time.

These dynamic Bayesian networks can then be used for updating the dynamic occupancy grids using the methodology described in the next section.

3.4 The Methodology for Navigation

The navigation system consists of two parts. The first is a motion planner that looks at the occupancy grids and determines the current state of the robot's neighborhood and then plans the next motion to execute. This motion planner can be dependent

on the environment in which the robot is operating. A fuzzy rule based system can also be used to select from a set of motion planners depending on the characteristics of the environment, thus achieving generalization in robot navigation.

The second part of the navigation system prescribes the methodology used in the framework to update the PDFs associated with the dynamic occupancy grids, based on the results of the dynamic Bayesian networks. This component system continuously reads the sensors and whenever any new information is obtained from a sensor, corresponding Bayesian networks that have that sensor as one of their nodes are started up in parallel. After belief propagation as described above, the new PDFs generated are used to update the corresponding PDFs in the dynamic occupancy grids. The Bayesian networks are fired up asynchronously, whenever any of the sensors generates some new information. The occupancy grids are also updated in an asynchronous manner. This is especially useful since the motion planner does not have to wait for the occupancy grid to be updated but simply uses the current PDFs to produce an estimate of the environment.

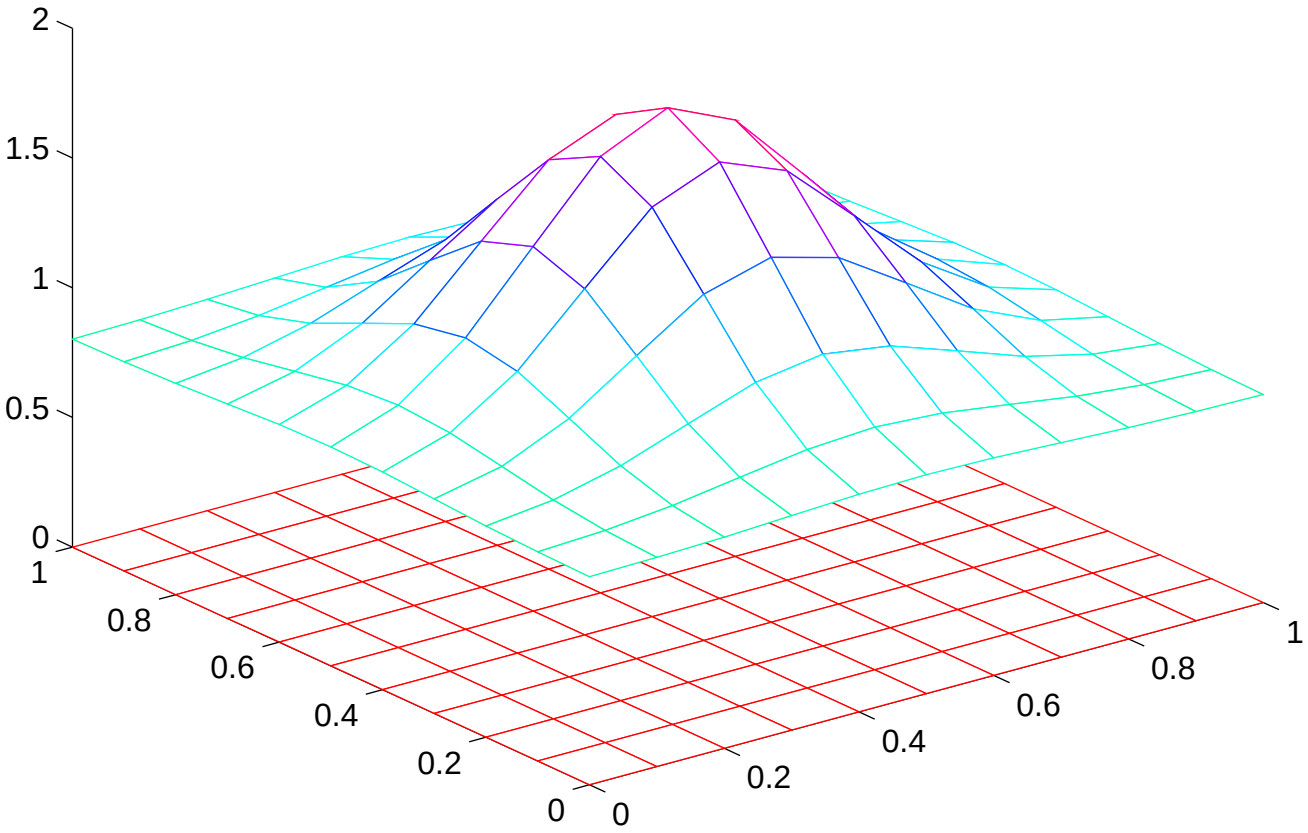

Figure 3.1: A "Fat" PDF showing more uncertainty in estimate due to lack of updates.

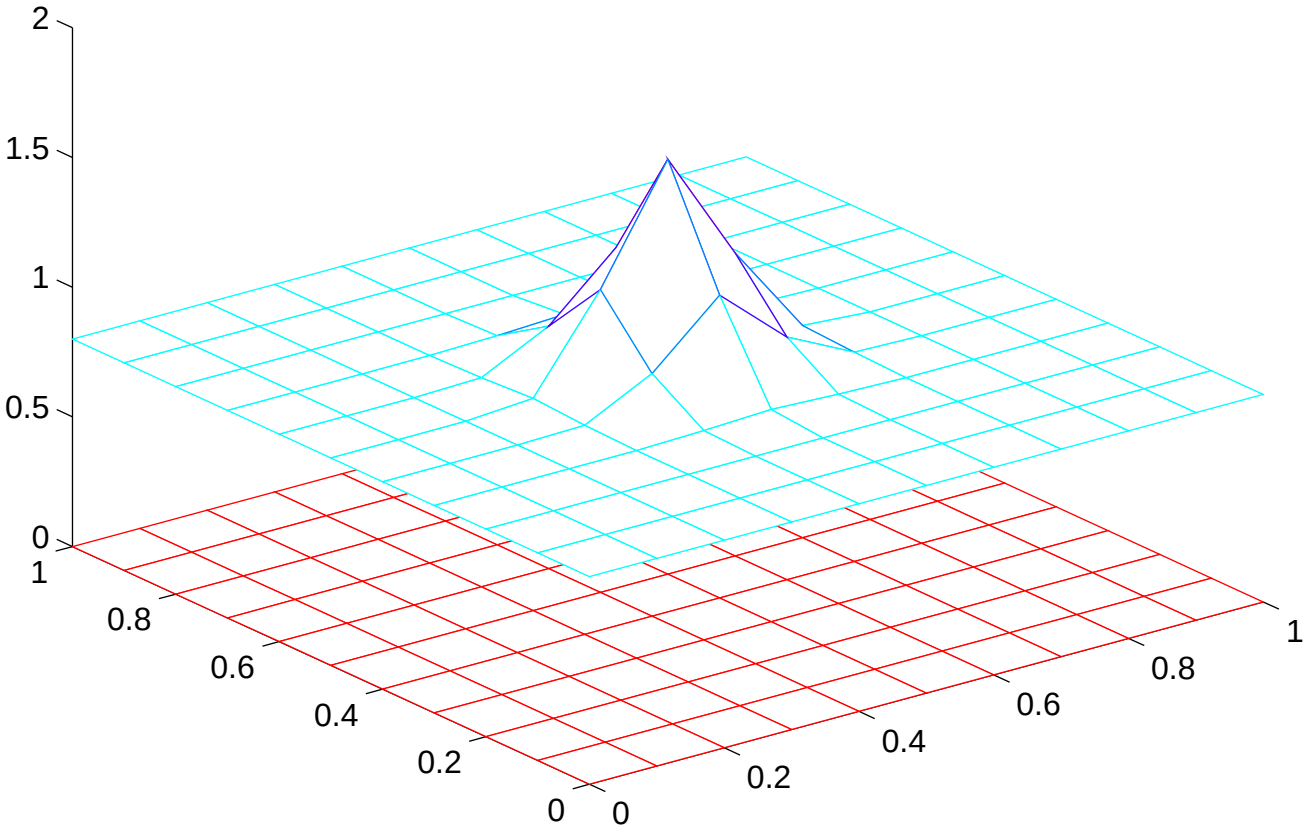

Figure 3.2: A "Thin" PDF showing less uncertainty in estimate due to frequent updates.

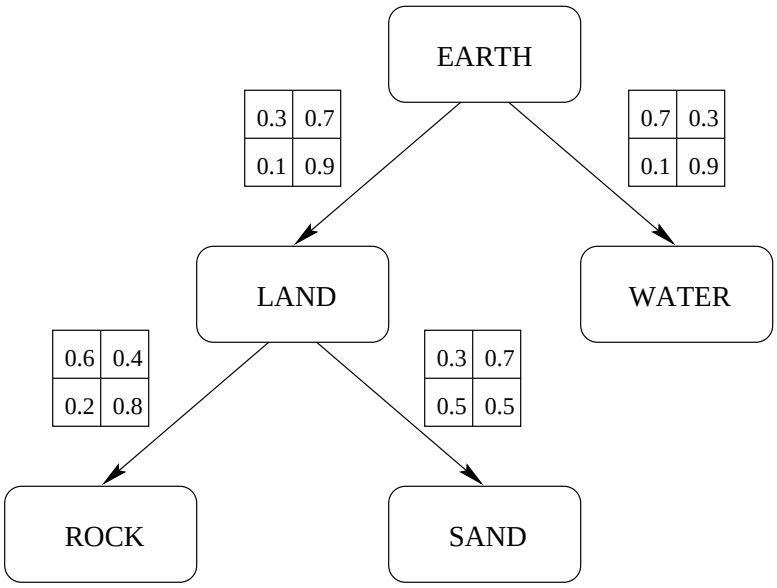

Figure 3.3: An example of a Bayesian Network

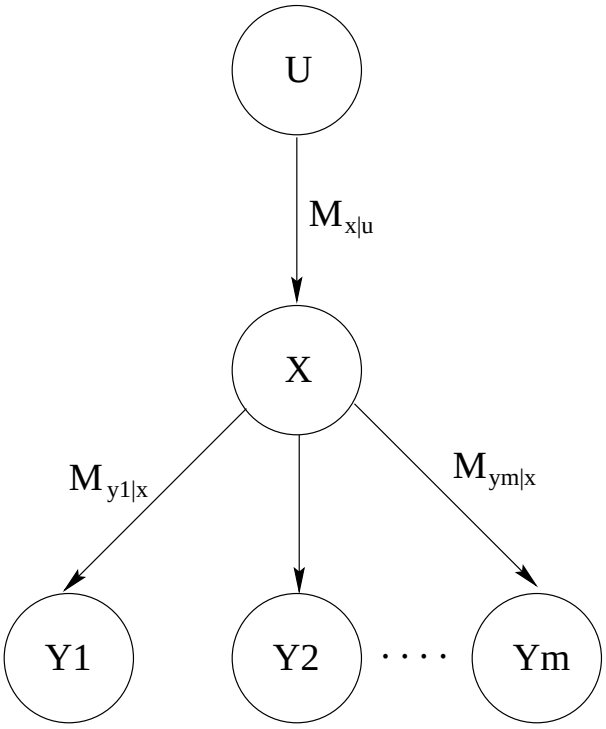

Figure 3.4: A Typical node X of a Generalized Bayesian Network.

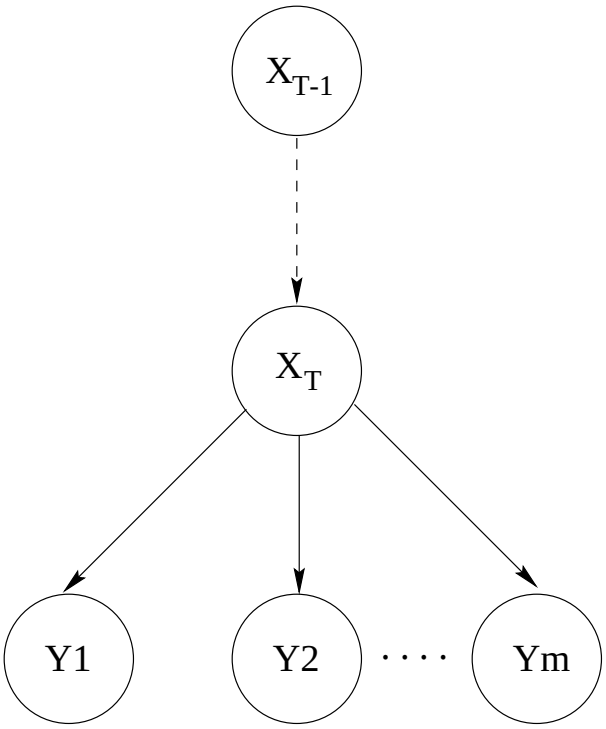

Figure 3.5: Incorporating past information into a Bayesian network using a history node.

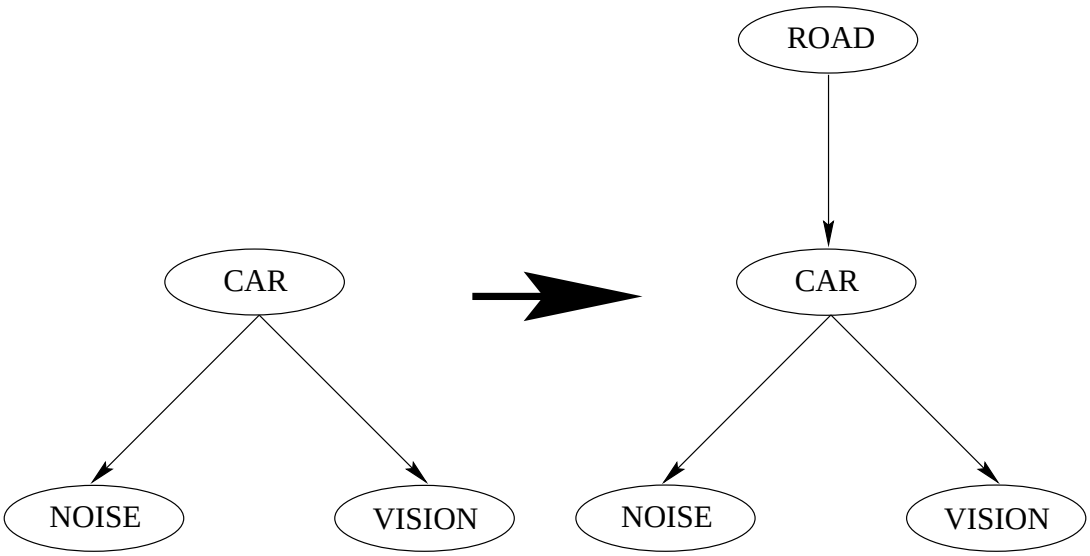

Figure 3.6: A dynamic Bayesian network for car and road detection from vision and noise sensors. The road detection node is adaptively added to the original car detection Bayesian network when the probability of the presence of car gets more than 0.5.

Chapter 4

Experimental Research Effort

We have recently acquired two mobile wheelchair robots with limited *off-road* navigation capabilities. These will serve as a test-bed for our framework in the real world. More details of the specifics of the wheelchair robots can be found in Appendix B. We are also in the process of building virtual environments on an SGI machine for use as simulator test-beds to model multiple complex test environments.

We are currently involved in two major research projects related to the development of our proposed framework. The first research effort was performed as part of the Mobile Robot practicum offered in the Spring semester of 1997 and aimed at developing and implementing an obstacle avoidance system for the wheelchair robots.

The second research effort is aimed at developing an efficient, platform independent parallel implementation of the memory-based object recognition algorithm proposed by Nelson. We describe our recent successes and failures in both of these efforts and highlight some of our future research plans for the next year.

4.1 Obstacle Avoidance

The goal of the obstacle avoidance research effort was to implement a reactive obstacle avoidance strategy for allowing the wheelchair to perform collision free navigation in tight and cramped spaces. We have developed an ad-hoc rule-based reasoning strategy that uses the various sensors (bump sensors, infra-red proximity sensors and sonar sensor) on the wheelchair to get information about obstacles in the neighborhood of the robot. The obstacle avoidance algorithm then takes the motor control commands received from the navigation system (human using joystick or autonomous driving program) and runs them through the rule database, adjusting their values depending on the locations of the obstacles before sending them on to the onboard motor controller. The current obstacle avoidance algorithm has the following rule base:

```
if obstacle detected in front of robot then
  if back is clear and both corners are not
     then reverse at low speed
  else if one corner is clear
     then move forward at low speed while turning moderately
          towards clear corner
  else if one side is clear
     then move forward at low speed while turning sharply
          towards clear side
  else
     turn sharply towards one side
else
  if sonar reading < 0.7 m
     then slow to 20% of maximum speed
  else if sonar reading < 1.2 m
     then slow to 50% of maximum speed
  if one corner is not clear
     then turn sharply away from that corner
  else if one side is not clear
     then turn slightly away from that side
  else if drive command is to turn towards a side
     then if corner is clear
        turn slightly away from that side
     else turn sharply away from that side
  else if drive command is to go in reverse
     then if back is not clear
        then if side is clear
           turn in place to clear side
     else turn in place to any side
  else if driving command higher than maximum speed
     limit driving command to maximum speed
```

The above obstacle avoidance algorithm was implemented using ARC, a software development system for the wheelchair robot, and integrated into a remote operation demonstration system with the help of Chris Eveland. Multiple tests were performed on this system for navigation through cramped doorways, computer lab environments, among moving people and in hallways. The tests showed that the system is extremely robust when the size of the obstacles is large enough to be detected fairly easily by the infrared proximity detectors. Our obstacle avoidance algorithm performs very well on very hard tasks such as going through a small doorway but fails to follow

walls well. Part of this is due to a control problem that is causing the robot to drift to the right.

Another aspect of this research work involved obtaining a better configuration of the Infra Red proximity sensors on the robot. The original configuration was found to be lacking and left major gaps and holes where obstacles were not being detected. We modified the placement of these sensors based on experimental results and obtained a better coverage of the wheelchair's environment. Our next step is to model the sensors in the simulator and find their optimal spatial configuration for obstacle avoidance.

Timing tests were run with the help of Craig Harmon on the wheelchairs when it was performing obstacle avoidance. It was determined that the obstacle avoidance program was generating drive commands to the motor controllers at a rate of 60-80 Hz per second. This rate is very good and reflects our extremely simplistic approach to the problem. Some of the limitations of the current system include the inability to detect thin vertical objects such as chair and table legs, and dark color objects due to sensor limitations. Also, the approach is completely ad-hoc and all refinement until now has been performed on the basis of empirical results.

We propose to continue research in obstacle avoidance and formalize the above system into a fuzzy rule based logic controller. This formalization would help us better understand the operation of our system, while making it modular and easily maintainable. We plan to use this obstacle avoidance algorithm in conjunction with a path planner to perform motion planning for the wheelchair.

4.2 Object Recognition

The second research effort is being done in collaboration with Randal Nelson and is aimed at developing a platform independent and efficient parallel implementation of Nelson's memory-based object recognition algorithm. This research work has three major goals:

1. Develop an efficient parallel implementation of the object recognition algorithm to reduce the time required for object recognition.

2. Showcase the abilities of the TreadMarks and Cashmere Protocols developed for Distributed Shared Memory systems by running the parallelized object recognition algorithm on a cluster of DEC Alphas

3. Develop a client-server protocol to allow the wheelchair to communicate images to a high-end vision server for fast landmark detection using the parallel object recognition algorithm

We have achieved moderate success in our goals until now and hope to achieve complete success shortly. We have developed a client server protocol that is capable of transporting images from the computer on board the wheelchair, via a wireless ethernet connection, to a remote server that can run the object recognition algorithm and return the results to the wheelchair.

We have also made much headway into developing an efficient parallel implementation of the object recognition algorithm. Simple parallelization and message passing strategies have been studied in depth and partly implemented before being discarded due to unforeseen issues with shared memory architecture behavior. Our current approach is based on a platform independent shared memory allocator that is under the process of being implemented. The actual routines performing the object recognition have been parallelized and can be tested soon as this shared memory allocator is fully implemented and tested.

The design approach used for the system breaks the object database into multiple local memories that are shared between a central process and the local process. Each local process is given the set of keys on which the search has to be performed and they all look into their portions of the database in parallel, generating hypotheses about possible object models and storing them in fully shared memory (among all processes). The central process then selects the best hypothesis and outputs that as the identity and pose estimation of the object in the source image. This portion of the object recognition algorithm has been implemented but has not yet been tested due to a problem with allocating and deallocating shared memory segments. We hope to fix this memory allocator problem soon and perform testing on the performance improvements of the parallelization task.

4.3 Future Work

The other major research accomplished so far is the theoretical development of the generalized framework for autonomous mobile robot navigation. Since we have been able to establish a very strong mathematical basis for our framework approach, we believe it can be implemented and should provide us with good performance. Our future research efforts are aimed at implementing dynamic occupancy grids and dynamic Bayesian networks on the wheelchair robots and using these to perform experimentation on the performance of robot navigation, both in the real world and in the simulated world.

We have chosen a set of three environments that we wish the robot to perform robust navigation in. These are:

1. Navigating in structured, dynamic, and known outdoor terrain (specifically navigating an autonomous robot from one end of the university campus to another).

2. Navigating in unstructured, dynamic, and unknown outdoor terrain (specifically navigating through a wooded area adjoining the university campus).

3. Navigating in an indoor environment in the presence of predictable dynamic objects (specifically navigating through the hallways and corridors of the university buildings).

These three tasks represent three very different environments. We aim to study the performance of our proposed approach using the above test-beds and analyze the generalization observed.

Chapter 5

Conclusions

We have aimed to acquaint the reader with the issues faced in autonomous mobile robot navigation. The autonomous mobile robot navigation problem can be decomposed into smaller subproblems that deal with robot localization, goal specification, goal recognition, motion planning, sensor fusion etc. Various methods and techniques have been proposed for solving these problems in different environments.

The complexity of the navigation problem is increased when uncertainties are generated in the environment due to a variety of reasons. Some of these include (1) the presence of dynamic obstacles, (2) lack of complete a-priori knowledge about the environment, (3) the design weaknesses and unreliability of sensing equipment, and (4) the presence of multiple, possibly conflicting, sensors. Current techniques fail to perform robot navigation in minimally constrained environments given uncertainties arising due to the above mentioned factors..

A survey of the vast body of literature in mobile robotics leads us to conclude that while many techniques that perform robust robot navigation in appropriately constrained environments exist, none of these are obviously extendible to a relaxation of the associated constraints. Another problem with the research efforts has been the individualized nature and limited scope of the published work. Most researchers take a specialized sub-problem (such as robot localization) of the mobile robot navigation problem and then propose a solution for that. However, a higher level implementation that uses the proposed technique for solving the sub-problem in conjunction with other algorithms required for performing robust robot navigation is often missing. This has led to very little research work in the area of generalized autonomous mobile robot navigation.

We believe that while it is very hard (if not impossible) to find a single robust technique for robot navigation that will perform well in minimally constrained environments with varying characteristics, a generalized framework approach can select from a repertoire of such methods based on the environment the robot is currently

operating in. To accomplish this, we propose a probabilistic generalized framework that uses dynamic occupancy grids for representation of the environment characteristics and updates this representation using probabilistic data generated via dynamic Bayesian networks. We also provide a complete analysis of the theoretical foundations of this framework.

Our next task is to implement this framework for mobile robot navigation using the off-road wheelchairs at the University of Rochester. Some of our past research work has focused on creating obstacle avoidance algorithms that work well in highly cramped and completely unknown environments. We show the robustness of our technique via empirical studies that produce good results, especially for traditionally hard tasks such as going through a small doorway. We are also actively involved in developing an efficient parallel algorithm for fast object recognition. We aim to use this algorithm for performing landmark detection and robot localization on the wheelchairs.

Bibliography

[1] T. Aoki *et al.* Acquisition of Optimal Action Selection to Avoid Moving Obstacles in Autonomous Mobile Robot. *Proceedings of ICRA-96*, Minneapolis, MN, 22-28 Apr, 1996, pp 2055-60.

[2] D. Apostolopoulus, H. Schempf, and J. West. Mobile Robot for Automatic Installation of Floor Tiles. *Proceedings of ICRA-96*, Minneapolis, MN, 22-28 Apr, 1996, pp 3652-7.

[3] S. Azami, S. Katahara, and M. Aoki. Route Guidance Sign Identification using 2-D Structural Description. *Proceedings of the 1996 IEEE Intelligent Vehicles Symposium*, Tokyo, Japan, 19-20 Sep, 1996, pp 153-8.

[4] D.H. Ballard and C.M. Brown. *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[5] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*, Academic Press Limited, San Diego, CA, 1988.

[6] J. Borenstein, H.R. Everett, and L. Feng. *Navigating Mobile Robots : Systems and Techniques*, A K Peters Ltd., Wellesley, MA, 1996.

[7] H.R. Beom and H.S. Cho. A Sensor Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, Mar 1995, pp 464-77.

[8] B. Bouilly and T. Simeon. A Sensor Based Motion Planner for Mobile Robot Navigation with Uncertainty. *Reasoning With Uncertainty in Robotics 1995*, Amsterdam, The Netherlands, 4-6 Dec, 1995, pp 235-47.

[9] V. Brajovic and T. Kanade. Computational Sensors for Global Operations. *Proceedings of the 23rd Image Understanding Workshop*, Monterey, CA, 13-16 Nov, 1994, pp 621-30.

[10] V. Brajovic and T. Kanade. A Sorting Image Sensor : An Example of Massively Parallel Intensity-to-Time Processing for Low-Latency Computational Sensors. *Proceedings of ICRA-96*, Minneapolis, MN, 22-28 Apr, 1996, pp 1638-43.

[11] R.A. Brooks and J.H. Connell. Asynchronous Distributed Control System for a Mobile Robot. *Proceedings of SPIE*, vol. 727, Oct 1986, pp 77-84.

[12] C.M. Brown *et al.* Distributed Data Fusion Using Kalman Filtering : A Robotics Approach. *Data Fusion in Robotics and Machine Intelligence*, Academic Press Limited, San Diego, CA, 1992, pp 267-309.

[13] J. Canny. *The Theory of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.

[14] C.C. Chang, K.T. Song. Sensor-Based Motion Planning of a Mobile Robot in a Dynamic Environment. *Proceedings of the 1996 IEEE IECON*, Taipei, Taiwan, 5-10 Aug, 1996, pp 766-71.

[15] D. Chung and F.L. Merat. Neural Network Based Sensor Array Signal Processing. *Proceedings of MFI-96*, Washington, D.C., 1996, pp 757-64.

[16] I.J. Cox. Blanche - An Experiment in Guidance and Navigation of an Autonomous Mobile Robot. *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, 1991, pp 193-204.

[17] J.L. Crowley. Mathematical Foundations of Navigation and Perception For an Autonomous Mobile Robot. *Reasoning With Uncertainty in Robotics 1995*, Amsterdam, The Netherlands, 4-6 Dec, 1995, pp 9-51.

[18] E.D. Dickmanns and N. Muller. Scene Recognition and Landmark Navigation for Road Vehicles. *Proceedings 2nd IFAC Conference on Intelligent Autonomous Vehicles*, Espoo, Finland, 12-14 June, 1995, pp 199-204.

[19] A. Dubrawski and J.L. Crowley. Self-Supervised Neural System for Reactive Navigation. *Proceedings 1994 IEEE International Conference on Robotics and Automation*, San Diego, CA, 8-13 May, 1994, pp 2076-81.

[20] A. Elfes. Occupancy Grids : A Probabilistic Framework for Robot Perception and Navigation. *Ph.D. thesis*, Carnegie Mellon University, Pittsburgh, PA, May 1989.

[21] A. Elfes. Robot Navigation: Integrating Perception, Environmental Constraints and Task Execution Within a Probabilistic Framework. *Reasoning with Uncertainty in Robotics. International Workshop, RUR'95 Proceedings*, Amsterdam, Netherlands, 4-6 Dec, 1995, pp 93-129.

[22] A. de la Escalera et. al. Neural Traffic Sign Recognition for Autonomous Vehicles. *Proceedings of IECON'94*, Bologna, Italy, 5-9 Sept, 1994, pp 841-6.

[23] K. Eshraghian. Challenges in Future Technologies. *Proceedings of the 14th IEEE VLSI Test Symposium*, Princeton, NJ, 28 Apr-1 May, 1996.

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

[25] C. Fennema and A.R. Hanson. Experiments in Autonomous Navigation. *Proceedings of the IEEE International Workshop on Intelligent Motion Control*, Istanbul, Turkey, 20-22 Aug, 1990, pp 29-37.

[26] T. Fraichard and C. Laugier. Dynamic Trajectory Planning, Path-Velocity Decomposition and Adjacent Paths. *Proceedings of IJCAI-93*, Chambery, France, 28 Aug-3 Sep, 1993, pp 1592-7.

[27] K. Fujimura. *Motion Planning in Dynamic Environments*, Springer-Verlag, Tokyo, Japan, 1991.

[28] R.F. Gans. Following Behavior Using Moving Potential Fields. *Computer Science Technical Report TR 603*, University of Rochester, Rochester, NY, Jan 1996.

[29] R. Greiner and R. Isukapalli. Learning to Select Useful Landmarks. *IEEE Transactions on Systems, Man, and Cybernetics - Part B : Cybernetics*, vol. 26, no. 3, June 1996, pp 437-449.

[30] W.E.L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, MA, 1990.

[31] R. Gutsche, C. Laloni and F.M. Wahl. Path Planning for Mobile Vehicles Within Dynamical Worlds Using Statistical Data. *Proceedings of IROS '94*, Munich, Germany, 12-16 Sep, 1994, pp 454-61.

[32] G. Hager, M. Mintz. Searching for Information. *Proceedings of the AAAI Workshop on Spatial Reasoning and Multisensor Fusion*, Los Altos, CA, 5-7 Oct, 1987, pp 313-22.

[33] A.B. Harvey. New Directions at NSF. *Proceedings of the SPIE*, vol. 2527, July 1995, pp 370-74.

[34] R.E. Helmick, J.E. Conte, and T.R. Rice. Absolute Sensor Alignment using GPS. *Proceedings of the SPIE*, vol. 2739, 1996, pp168-79.

[35] R. Jarvis. An All-Terrain Intelligent Autonomous Vehicle With Sensor-Fusion-Based Navigation Capabilities. *Control Engineering Practice*, vol. 4, no. 4, Apr 1996, pp 481-6.

[36] T. Jochem and D. Pomerleau. Life in the Fast Lane: The Evolution of an Adaptive Vehicle Control System. *AI Magazine*, vol. 17, no. 2, 1996, pp 11-50.

[37] L. Kari. DNA Computers, Tomorrow's Reality. *Bulletin of the European Association for Theoretical Computer Science*, no. 59, June 1996, pp 256-66.

[38] G. Kamberova, M. Mintz. Robust Multisensor Fusion : A Decision Theoretic Approach. *Proceedings of the 1990 DARPA Image Understanding Workshop*, 1990, pp 867-73.

[39] M. Kam, X. Zhu and P. Kalata. Sensor Fusion for Mobile Robot Navigation. *Proceedings of the IEEE*, vol. 85, no. 1, Jan 1997, pp 108-19.

[40] K. Kant and S.W. Zucker. Toward Efficient Trajectory Planning: The Path-Velocity Decomposition. *International Journal of Robotics Research*, vol. 5, no. 3, Fall 1986, pp 72-89.

[41] K. Kluge and C. Thorpe. The YARF System for Vision-Based Road Following. *Mathematical and Computer Modeling*, vol. 22, no. 4-7, Aug 1995, pp 213-33.

[42] W.S. Ko, L.D. Senevieatne, and S.W.E. Earles. Space Representation and Map Building – A Triangulation Model to Path Planning with Obstacle Avoidance. *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, 26-30 July, 1993, pp 2222-7.

[43] D. Kortenkamp et. al. Mobile Robot Exploration and Navigation of Indoor Spaces Using Sonar and Vision. *Proceedings of CIRFFSS'94*, Houston, TX, 21-24 Mar, 1994, pp 509-19.

[44] S. Kristensen. Sensor Planning with Bayesian Decision Theory. *Reasoning with Uncertainty in Robotics. International Workshop, RUR'95 Proceedings*, Amsterdam, Netherlands, 4-6 Dec, 1995, pp 353-67.

[45] S. Kristensen and H.I. Christensen. Decision-Theoretic Multisensor Planning and Integration for Mobile Robot Navigation. *Proceedings of MFI-96*, Washington, D.C., 1996, pp 517-24.

[46] J.C. Latombe. *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[47] J.C. Latombe. Robot Algorithms. *Workshop on the Algorithmic Foundations of Robotics*, San Francisco, CA, 17-19 Feb, 1994, pp 1-19.

[48] J. Leonard and H.F. Durrant-Whyte. Mobile Robot Localization by Tracking Geometric Beacons. *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, 1991, pp 376-82.

[49] L.F. Lee, A. Kean. An architecture for autonomous flying vehicles: a preliminary report. *Proceedings of PRICAI'96*, Cairns, Australia, 26-30 Aug, 1996, pp 613-24.

[50] W. Li. Perception-Action Behavior Control of a Mobile Robot in Uncertain Environments Using Fuzzy Logic. *Proceedings of IROS-94*, Munich, Germany, 12-16 Sep, 1994, pp 439-46.

[51] C.C. Lin and R.L. Tummala. Mobile Robot Navigation using Artificial Landmarks. *Journal of Robotic Systems*, vol.14, no. 2, Feb 1997, pp 93-106.

[52] R.C. Luo, M.G. Kay. Data Fusion and Sensor Integration : State-of-the-Art 1990s. *Data Fusion in Robotics and Machine Intelligence*, Academic Press Limited, San Diego, CA, 1992, pp 7-136.

[53] R.C. Luo, H. Potlapalli and D.W. Hislop. Outdoor Landmark Recognition Using Fractal Based Vision and Neural Networks.

[54] R.C. Luo and H. Potlapalli. Landmark Recognition Using Projection Learning for Mobile Robot Navigation. *Proceedings of the 1994 IEEE International Conference on Neural Networks*, Orlando, FL, 27 June-2 July, 1994, pp 2703-8.

[55] S. Maeyama, N. Ishikawa and S. Yuta. Rule Based Filtering and Fusion of Odometry and Gyroscope for a Fail Safe Dead Reckoning System of a Mobile Robot. *Proceedings of MFI-96*, Washington, D.C., 8-11 Dec, 1996, pp541-8.

[56] M.J. Mataric. Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, nos. 2-3, 1997.

[57] A.L. Meyrowitz, D.R. Blidberg, and R.C. Michelson. Autonomous Vehicles. *Proceedings of the IEEE*, vol. 84, no. 8, Aug 1996, pp 1147-1164.

[58] H. Murase and S.K. Nayar. Visual Learning and Recognition of 3D Objects from Appearance. *International Journal of Computer Vision*, vol. 14, no. 1, Jan 1995, pp 5-24.

[59] Y.S. Nam, B.H. Lee, and M.S. Kim. View-Time Based Moving Obstacle Avoidance Using Stochastic Prediction of Obstacle Motion. *Proceedings. IEEE International Conference on Robotics and Automation*, Minneapolis, MN, 22-28 Apr, 1996, pp 1081-6.

[60] S.K. Nayar, S.A. Nene and H. Murase. Real-Time 100 Object Recognition System. *Proceedings of the 1996 Image Understanding Workshop*, Palm Springs, CA, 12-15 Feb, 1996, pp 1223-7.

[61] R.C. Nelson. Memory-Based Recognition for 3-D Objects. *Proceedings ARPA Image Understanding Workshop*, Palm Springs, CA, Feb 1996, pp 1305-1310.

[62] R.C. Nelson. Experiments on (Intelligent) Brute Force Methods for Appearance-Based Object Recognition. *1997 DARPA Image Understanding Workshop*, New Orleans, LA, May 1997, to appear.

[63] J.T.Nielson. CALCM - The Untold Story of the Weapon Used to Start the Gulf War. *IEEE Aerospace and Electronic Systems Magazine*, vol. 9, no. 7, Jul 1994, pp 18-22.

[64] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* 2nd ed., Morgan Kaufmann Publishers, San Francisco, CA, 1988.

[65] N.S.V. Rao. Robot Navigation in Unknown Generalized Polygonal Terrains Using Vision Sensors. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 6, pp 947-62.

[66] N.S.V. Rao. Distributed Decision Fusion Using Empirical Estimation. *Proceedings of MFI-96*, Washington, D.C., 8-11 Dec, 1996, pp 697-704.

[67] J. Reif and M. Sharir. Motion Planning in the Presence of Moving Obstacles. *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, Portland, OR, 1985, pp 144-54.

[68] M. Sekiguchi, H. Okada and N. Watanabe. Neural Network Based Landmark Detection for Mobile Robot. *Proceedings of the SPIE*, vol. 2760, Apr 1996, pp 216-23.

[69] F. Shibata *et al.* Mobile Robot Navigation by User-Friendly Goal Specification. *Proceedings. 5th IEEE International Workshop on Robot and Human Communication*, Tsukuba, Japan, 11-14 Nov, 1996, pp 439-44.

[70] R. Smith and P. Cheeseman. On the Estimation and Representation of Spatial Uncertainty. *International Journal of Robotics Research*, vol. 5, no. 4, Winter 1987, pp 56-68.

[71] S.H. Suh *et al.* CAD-MAP and Estimation of ALV Positions in Mountainous Areas. *Robotica*, vol. 12, pt. 4, pp 287-97.

[72] K.T. Sutherland and W.B. Thompson. Inexact Navigation. *Proceedings of ICRA-93*, Atlanta, GA, 2-6 May, 1993, pp 1-7.

[73] H. Suzuki and A. Arimoto. A Recursive Method of Trajectory Planning for a Point-Like Mobile Robot in Transient Environment Utilizing Paint Procedure. *Journal of the Robotics Society of Japan*, vol. 8, 1990, pp 9-19.

[74] S. Thrun and A. Bucken. Learning Maps for Indoor Mobile Robot Navigation. *Technical Report CMU-CS-96-121*, Carnegie Mellon University, Pittsburgh, PA, 1996.

[75] S, Thrun. A Bayesian Approach to Landmark Discovery and Active Perception in Mobile Robot Navigation. *Technical Report CMU-CS-96-122*, Carnegie Mellon University, Pittsburgh, PA, 1996.

[76] W.B. Thompson *et al.* Geometric Reasoning for Map-Based Localization. *Computer Science Technical Report UUCS-96-006*, University of Utah, Salt Lake City, UT, May, 1996.

[77] T. Tsubochi, A. Hirose, and S. Arimoto. A Navigation Scheme with Learning for a Mobile Robot Among Multiple Moving Obstacles. *Proceedings of IROS '93. Intelligent Robots for Flexibility*, p. 3, vol. 2317, 1993, pp 2234-2240.

[78] P. Wide and D. Driankov. A Fuzzy Approach to Multi-Sensor Data Fusion for Quality Profile Classification. *Proceedings of MFI-96*, Washington, D.C., 1996, pp 215-21.

[79] C.H. Wu. A Coordinated Motion Planner Among Moving Machines and Objects. *Proceedings of IEEE Symposium on Intelligent Vehicles 1995*, New York, NY, 25-26 Sep, 1995, pp 188-93.

[80] B. Yamauchi and R. Beer. Spatial Learning in Dynamic Environments. *IEEE Transactions on Man, Systems and Cybernetics*, vol. 26, no. 3, Jun 1996, pp 496-505.

[81] S. Yuta. Experimental Research on the Autonomous Mobile Robot Which Navigates in the Real World. *Proceedings of the Second International Conference on Mechatronics and Machine Vision in Practice*, Hong Kong, 12-14 Sept, 1995, pp 59-66.

Appendix A

Triangulation of Active Beacons for Localization

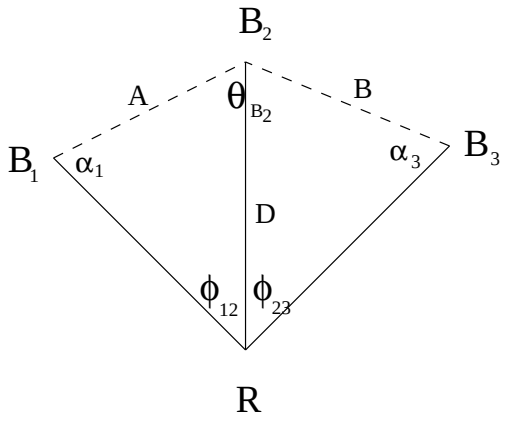

Figure A.1: Triangulation Using Three Active Beacons

Suppose the robot R currently has three beacons B_1, B_2, and B_3 in its field of *view*. Using the radio signal transmitted from the beacons, the robot can determine the measurements of the angles ϕ_{12} and ϕ_{23}. To determine the position and orientation of the robot, the robot needs to determine the distance D to one of the beacons, say B_2. The robot also has a-priori knowledge about the distances A and B between the beacons and the angular separation of the three beacons θ_{B_2}. We can compute this distance D as follows :

First, note that $\alpha_1 + \alpha_3 + \theta_{B_2} + \phi_{12} + \phi_{23} = 360^o$. Let us rewrite this as $\alpha_1 = c - \alpha_3$ where $c = 360^o - \theta_{B_2} - \phi_{12} - \phi_{23}$ and is known from sensor observations and a-priori information.

Now, using the trigonometric *law of sines*

$$\frac{A}{\sin\phi_{12}} = \frac{D}{\sin\alpha_1} \tag{A.1}$$

$$\frac{B}{\sin\phi_{23}} = \frac{D}{\sin\alpha_3} \tag{A.2}$$

Using equations A.1 and A.2 and equating for D, we get

$$\begin{aligned}
\frac{A\sin\alpha_1}{\sin\phi_{12}} &= \frac{B\sin\alpha_3}{\sin\phi_{23}} \\
\frac{\sin\alpha_1}{\sin\alpha_3} &= \frac{B\sin\phi_{12}}{A\sin\phi_{23}} \\
\frac{\sin(c-\alpha_3)}{\sin\alpha_3} &= \frac{B\sin\phi_{12}}{A\sin\phi_{23}} \\
\frac{\sin c\cos\alpha_3 - \cos c\sin\alpha_3}{\sin\alpha_3} &= \frac{B\sin\phi_{12}}{A\sin\phi_{23}} \\
\frac{\sin c\cos\alpha_3}{\sin\alpha_3} - \cos c &= \frac{B\sin\phi_{12}}{A\sin\phi_{23}} \\
\frac{\cos\alpha_3}{\sin\alpha_3} &= \frac{1}{\sin c}\left(\frac{B\sin\phi_{12}}{A\sin\phi_{23}} + \cos c\right) \\
\alpha_3 &= \cot^{-1}\left[\frac{1}{\sin c}\left(\frac{B\sin\phi_{12}}{A\sin\phi_{23}} + \cos c\right)\right]
\end{aligned} \tag{A.3}$$

We can substitute the value of α_3 from equation A.3 into equation A.2 and solve for D as follows

$$D = \frac{B\sin\alpha_3}{\sin\phi_{23}} \tag{A.4}$$

Appendix B

Hardware for Proposed Experimentation

In addition to a virtual simulator under design on an SGI machine, we also possess two mobile off-road wheelchair robots. These robots are modified standard electric wheelchairs and run on two 12 volt batteries. There is an onboard 68332 microprocessor connected directly to the wheelchair motors and some of the sensors. This microprocessor is also connected via a serial line to a dual-Pentium computer that has been added to the wheelchair.

Besides the computing power of the 68332 and the dual-Pentium computers, a wireless ethernet connection hooks the wheelchair robot to the computer science department's network of computers. In addition to the computing hardware, the wheelchairs are also equipped with a range of sensors. Figure B.1 shows the spatial arrangements of various sensors on-board the wheelchair. There are two bump-sensors installed in front of the wheelchair that act as the last line of defense. An array of twelve infra-red proximity sensors are used to detect obstacles in the path of the wheelchair and a sonar sensor is used to gauge the distance to the nearest object in front of the wheelchair. All of the above sensors are directly connected to the 68332 microprocessor.

We have also installed a camera connected to the dual-Pentium computer onboard the wheelchair. The camera is installed facing forwards and is currently being used for remote operation of the wheelchair. The images from the camera are transferred via a TV antenna to a display monitor at a user station. We are currently working on integrating the camera with the other sensors to perform some vision related sensing tasks.

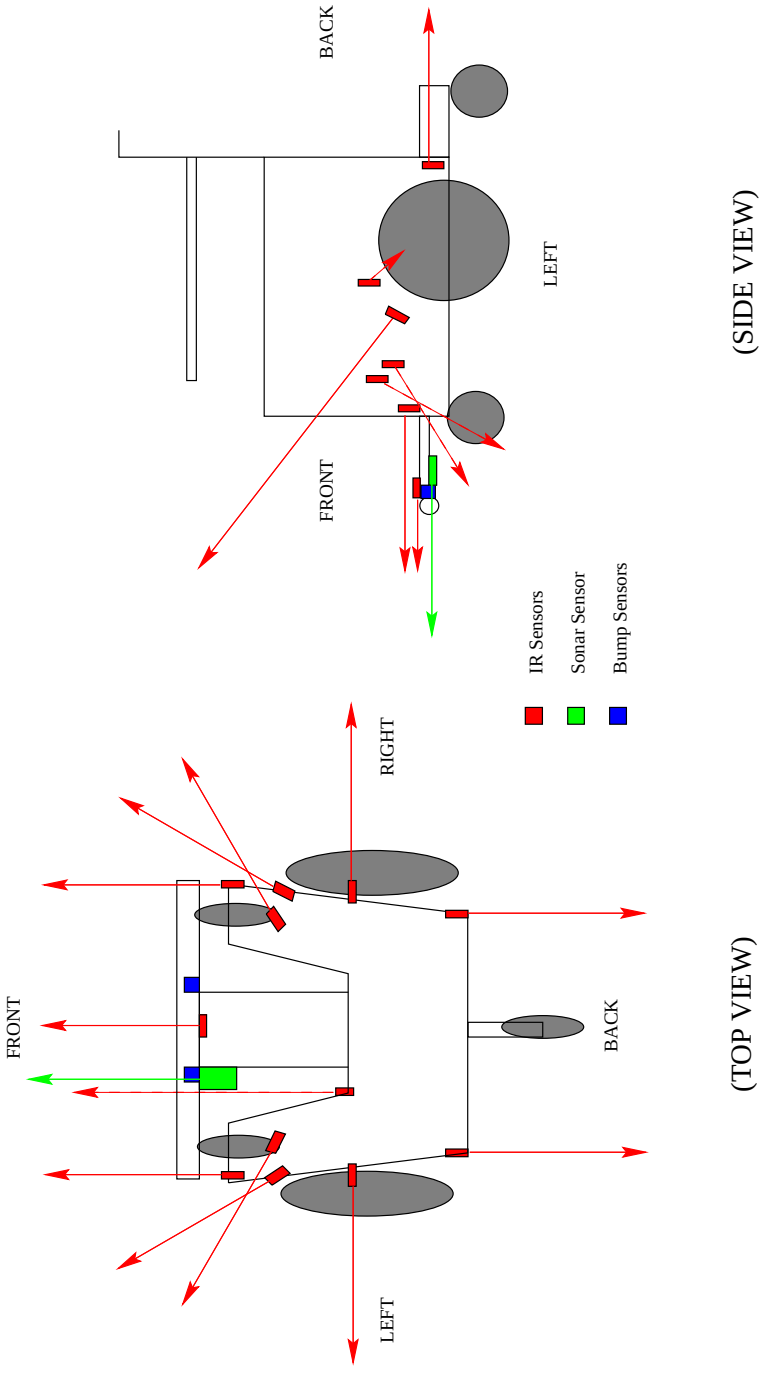

Figure B.1: Sensor Placement for Wheelchair.