The Design and Implementation of Alan Touring, the Campus Tourbot

by

Collin Eugene Johnson

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

Author . . . . . .
Department of Electrical Engineering and Computer Science
August 29, 2008

Certified by . . . . . .
Cynthia Breazeal
Associate Professor
Thesis Supervisor

Accepted by . . . . . .
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

The Design and Implementation of Alan Touring, the Campus Tourbot

by

Collin Eugene Johnson

Submitted to the Department of Electrical Engineering and Computer Science
on August 29, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Alan Touring, the Campus Tourbot, is a robotic tour-guide/tourist robot for parts of the MIT campus. Work on the robot began in January 2006, and for the past 31 months, I have devoted thousands of hours to designing, implementing, and testing the robot. This thesis describes the hardware and software systems created for the robot, performs an evaluation of the robot's functionality, and discusses the public's reactions to the robot when they encountered it driving through their world. As with any robotic system, the development of the Tourbot involved trial-and-error. A critique of some major design decisions that were made and how these decisions affected the development of the robot is performed. Suggestions for how the development process could have been improved based on lessons that I learned are then offered.

Thesis Supervisor: Cynthia Breazeal
Title: Associate Professor

Acknowledgments

First and foremost, I would like to thank the members of the Tourbot team, Will, Patrick, Behram, Jason, Jeremy, Kyle, and Mark, for all of their help in getting the robot built and running. I would especially like to acknowledge Will Bosworth and Patrick Barragan for sticking with the project from the beginning and dedicating their limited time to keeping the robot hardware functioning. Further thanks go to the MIT/Microsoft Alliance's iCampus program that provided the initial funding for the project, and the Edgerton Center, especially Sandi Lipnoski, for further support in handling the team's finances.

Cynthia has been a great adviser. She has help me in dealing with the calamities that are inherent in any robotics project. Her insight has really extended my thinking on the social possibilities of robots. Prof. Seth Teller and Prof. Daniela Rus provided much of my inspiration for pursuing robotics and have opened many doors for me as both a student and TA in their classes.

This thesis and my MIT career would not have happened without my incredible family. They have always been extremely supportive of my goals and desires which has allowed me to pursue all of my dreams. Hopefully Mom will forgive me for ignoring her requests to sleep more for five years, and Dad will forgive me being too busy for our annual Moab trip the past two years.

Fred Harris and Charlie Kohlhase have provided me with a wonderful music experiences for the past few years. Music was my escape valve during stressful times and has helped me keep my sanity while trying to finish my robot. Thanks to their support, music has become a fundamental part of my life.

Thanks also to Wayne for going well above and beyond the call of duty for a freshman adviser and being almost like family. All the lunches and discussions really helped solidify the person that I became while at MIT.

Lastly, I would like to thank Kinesis keyboards, Dell LCDs, RBH Sound speakers, Denon Electronics, caffeine, and all the wonderful musicians whose recordings I own for providing the perfect coding atmosphere in my Fortress of Solitude.

Contents

1 Introduction **13**

1.1 Overview 13

1.2 Thesis Outline 14

2 Hardware: The Brawn **17**

2.1 Chassis 17

2.2 Sensors 20

2.3 Computation and Display 21

2.4 Electrical System 22

3 Software: The Brains **25**

3.1 System Design 26

3.2 Communication System 26

3.2.1 DataManager 28

3.2.2 DataMinion 29

3.3 Low-level Interfaces 29

3.4 Localization 30

3.4.1 Odometry 31

3.4.2 Fiducial Recognition 31

3.5 Navigation 37

3.5.1 Mapping 38

3.5.2 Path Planning 44

3.5.3 Following the Path 49

3.6 Human Interface 51
3.6.1 Face Tracker 51
3.6.2 LCD Interface 52

4 System Evaluation **55**
4.1 Tourbot At the MIT Museum 55
4.1.1 Scenario 56
4.1.2 The Museum Environment 56
4.1.3 Evaluation 57
4.2 Reactions of the Public 63
4.2.1 Driving Down the Street 63
4.2.2 General Reactions 65
4.2.3 Appearance 66
4.2.4 Communication 67
4.2.5 Motion 70

5 Related Work **73**
5.1 RHINO and Minerva 73
5.2 CMU Mobots 75
5.3 RoboX 77
5.4 Robust Robots 79

6 Learning From Tourbot **81**
6.1 Hardware 81
6.1.1 Chassis 81
6.1.2 Sensors 86
6.2 Software 86
6.2.1 Communication 87
6.2.2 Localization 88
6.2.3 Navigation 90
6.2.4 Human Interactions 91

6.3 Lessons 94
6.3.1 The Team 94
6.3.2 The Design Process 96
6.3.3 Building A Strong Foundation 100
6.3.4 Spiraling 101
6.3.5 Using Math and Models 103

7 Future Work **105**
7.1 Robot Tourism 105
7.2 How Does Communication Affect the Perception of Robots? 107
7.3 Harassing Robots 108
7.4 Curious Robots 109
7.5 Making Robots Act Naturally In Crowds 113
7.6 Long-Term Deployment 115

8 Conclusions **119**

List of Figures

2-1 Side-by-side comparison of the original and final robot layouts. There was no cover around the base of the original robot. 18

2-2 An early rendering illustrating the need for the angled lidar 21

2-3 Electrical Connections for Tourbot 23

3-1 Organization of the software system 27

3-2 An Example of the Tourbot Communication System 28

3-4 Flow chart of data through the fiducial recognition system 32

3-3 A fiducial used by the robot 32

3-5 Comparsion of E_{min} and E_{max} for the fiducial shapes 34

3-6 A fiducial extracted from an image in the MIT Museum. Across the bottom of the image is the histogram of brightness values used for the threshold adjustment. Notice the very high peak corresponding to the ceiling color. 36

3-7 Flow of data through the navigation system 38

3-8 A stool that is invisible with the original 2D grid. Notice the much greater certainty that an object is in the path of the robot with the improved occupancy grid versus the original 2D grid. 40

3-9 An occupancy grid and cost map built by driving the robot around its lab. The curved hallway shows the robot's odometry error. Note that the vast majority of the world is considered untraversable due to the extra large configuration space. 42

3-10 Path generated by the local planner through a cost map. Notice how the edge falloff in the cost map keeps the robot equidistance from obstacles. 47
3-11 Trajectory followed by the robot along the path in Figure 3-10 50
3-12 Interface currently displayed on the touchscreen. 52
3-13 Display shown on the LCD. Bottom left corner is the face camera view. Bottom right is the ceiling camera view. 53

4-1 Frames that contain fiducials that are not recognized or are misread due to poor lighting. 58
4-2 The front and back of the robot. I leave it to the reader to decide which is which. 68

5-1 RHINO [2] 74
5-2 Minerva [26] 74
5-3 The CMU Mobots [31] 76
5-4 RoboX [30] 78

6-1 Vectors along which the robot would tip 83
6-2 Direct TCP/IP connections vs. UDP Multicasting 88
6-3 Valerie the Roboceptionist [13] 92
6-4 Tourbot team members (Left to Right): Behram Mistree, Collin Johnson, Patrick Barragan, Will Bosworth 95
6-5 Major classes in the communication system and their purposes 98

7-1 Layers of the spatial-semantic hierarchy. The symbolic layer is my proposed addition. 111
7-2 Flow chart for psychology-based robot motion 114

Chapter 1

Introduction

1.1 Overview

Human-robot interaction (HRI) is a rapidly expanding research field. HRI research aims to determine the factors that will allow robots to become a natural part of society. One can imagine many places where robots may become a part of society, but the most obvious centers around task-oriented robots capable of assisting humans in various daily tasks. The success of consumer robots like the Roomba demonstrates that there is a place for such robots in our society. The Roomba is a very simple robot though, and its interactions with humans are very minimal. Despite this, many people anthropomorphize the Roomba [7], giving it a name and speaking to it in a human manner. This result is exciting because, while the Roomba is not capable of real social interaction, the response it receives is very positive, making more complex and capable robots seem likely to be accepted into society.

The Campus Tour Bot, also known as Alan Touring, is a social robot designed to give tours of varying parts of the MIT campus. The robot provides a unique introduction to MIT. A typical tour of MIT by a human guide consists of a series of short stops at places of interest around campus where questions are answered. The Campus Tour Bot will support these interactions and will also incorporate visual information into its presentation. The largest difference between the robot and a human guide will be the range of tours that it will be able to provide. In particular,

the robot will not be able to travel up or down stairs and will be limited to indoor environments. Given the connected layout of MIT, though, this limitation does not significantly hamper the usefulness of the robot.

Tourbot operates in dynamic human environments and aims to replace or assist a human tour-guide. Because the robot is operating in a human environment, the goal of the project is to create a robot that behaves naturally in the presence of humans. For natural behavior, there are three means of creating a natural robot. First, the robot can be anthropomorphic, which makes the appearance of the robot human-like. Second, the robot can use human principles to move through the world, which makes the robot seem intelligent and aware of its environment. Finally, the robot can be socially intelligent by following societal norms during its interactions with people. The design of Tourbot focused on the robot moving naturally in the world. This design choice was based on the strengths of the team that built the robot. The team consists of engineers with more of a functional outlook than an artistic streak.

The robot has yet to give a tour of the main MIT campus, but has been involved in many activities at the MIT Museum. During the month of July 2008, the robot spent three to five days a week being tested and giving tours of the Robots and Beyond exhibit. People of all ages interacted with the robot. I accompanied the robot the entire time it was at the museum and talked with many people about the technical aspects of the robot, answered general questions about robotics, and discussed a variety of philosophical thoughts on technology in general. During this period, I learned many valuable lessons about the general public's expectations for a robot, as well as how a robot's design affects both its actual and perceived abilities.

1.2 Thesis Outline

The remainder of this thesis is organized as follows. Chapters Two and Three provide technical details on the hardware and software the were designed and implemented for the robot. Chapter Four is an evalution of my experiences in building and testing the robot, focusing on the robot's performance at the MIT Museum and the public's

reactions to the robot. Chapter Five describes previous tour-guide robots that have been successfully deployed at museums around the world. Chapter Six discusses lessons learned while building the robot, including what worked, what didn't, and the general engineering principles that, had they been applied, would have greatly benefitted the overall development. Chapters Seven and Eight conclude the thesis by providing future directions for research and summarizing the contributions of this thesis.

Chapter 2

Hardware: The Brawn

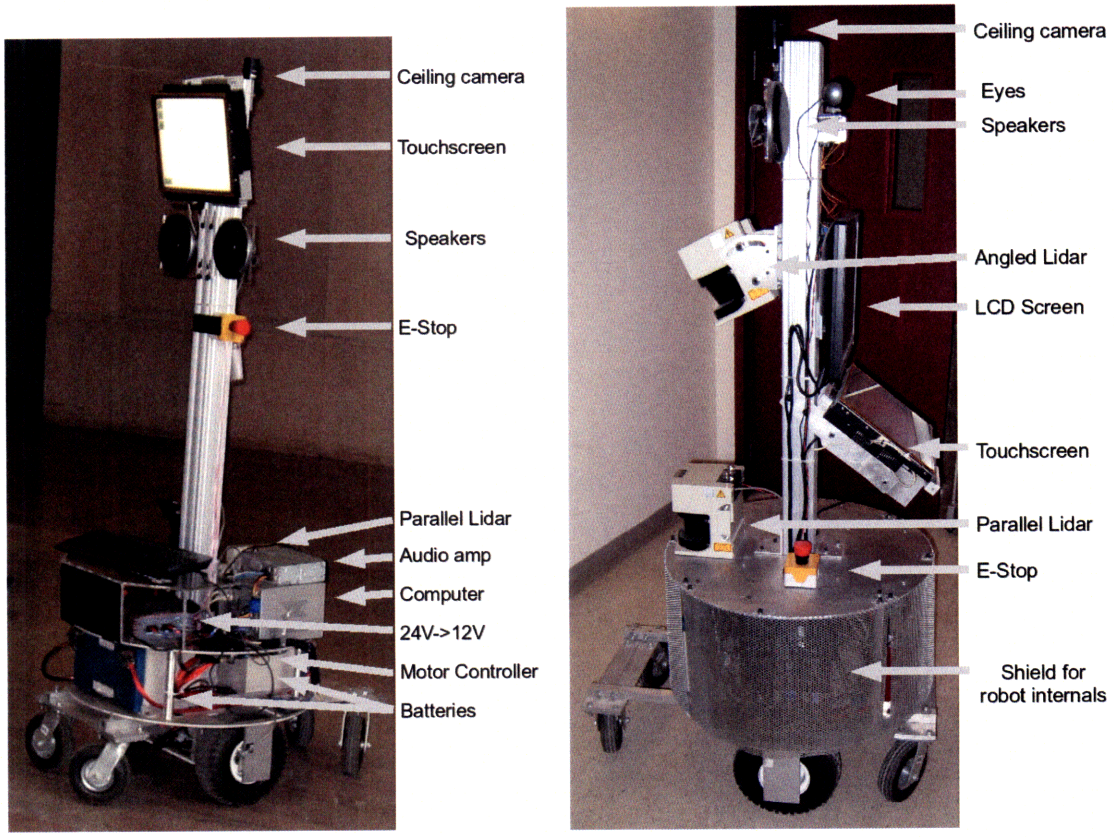
The robot is designed to operate indoors in buildings across the MIT campus. The robot is unable to traverse stairs or elevators, limiting his autonomous motion to a single floor of any building. The indoor constraint also limits the ability of the robot to withstand getting wet because waterproofing was not taken into account in the design of the robot. The robot needed to be large enough that it seemed substantial, but not infeasibly large for its environment.

In the initial design phase of the robot, a number of pre-built robot platforms were discussed, but the team ultimately decided to build the robot from scratch. The motivation for this decision was a general feeling among team members that you learn best by doing and buying a pre-built platform would defeat that purpose. Building a custom platform requires a great deal more effort and does not guarantee better results than the pre-built platform, but many of the difficult design decisions and problems that arise are invaluble experience for future projects.

The hardware for the robot was mostly constructed by other members of the Tourbot team, but I was involved in all discussions about the overall design.

2.1 Chassis

The design of the chassis aimed to create a robot that could be easily expanded with new hardware, while maintaining a slim enough profile to travel through standard



(a) Original Layout

(b) Final Layout

Figure 2-1: Side-by-side comparison of the original and final robot layouts. There was no cover around the base of the original robot.

doors. With this goal in mind, the initial robot platform consisted of a two circular plates with diameters of 0.66m mounted to a central spine of 80/20 about 1.8m tall. The robot used a two-wheel differential drive system with a caster wheel at the front and back of the robot. The aim was to keep all hardware within these circular plates to allow for simpler navigation. The two plates were 0.20m apart. The batteries, motors, and main fusebox were mounted to the bottom plate, while sensors and computers were mounted to the top plate. Figure 2-1 shows the original layout of the hardware. Wires ran through two holes drilled in the top plate and through the channels in the 80/20. The drive motors are NPC Robotics T64 with PT44 wheels with a diameter of 0.254m. The motors are powered by a Roboteq AX3500 motor controller.

After construction, two major flaws were discovered with this design. The foremost concern was that the robot could tip over with the given weight distribution and points of contact. The other major problem was that mounting sensors and computers on top of the second plate was dangerous to the robot and robot users because the electrical system of the robot was exposed to errant fingers and liquids. Both of these flaws were project-ending problems because they posed a major safety risk to people in the robot's environment. If the robot tipped, the central aluminum spine would move at high speeds in a chopping motion. If it hit a person, the aluminum could cause serious harm. Even without hitting a person, unacceptable damage to the robot's environment could result. A robot with accessible electrical connections poses a danger to people who might be curious about the robot, lean forward for a closer look, and then accidentally cause a short-circuit that could electrocute them. A falling object could similarly short-circuit the robot and destroy sensors or computers.

In addressing the tipping problem, a number of designs were considered: lowering the batteries, having a ballast that would tip to always keep the robot stable, or extending the caster wheels further from the drive wheels. Ultimately, it was chosen to extend the caster wheels further from the robot. In doing this, wheels with a larger diameter could be used, which allowed the robot to more easily travel across bumps several centimeters high that crop up at inopportune places across the MIT campus. In addition to extending the casters, twice as many casters were used, so that there are now two casters in both the front and back of the robot. The front casters were moved about 0.25m in front of the robot, and the back casters were extended about 0.05m behind the robot. These extensions broke one of the main goals of the initial design: keeping all hardware contained within the circle of the robot. However by this time, the time resources for most members of the team were very limited, so this was the most feasible solution from an implementation standpoint.

The solution to the electrial mounting problem was to raise the second plate 0.15m and mount all hardware on the underside of the plate. Moving the hardware and wiring to the underside of the robot had the additional benefit of cleaning up the overall appearance of the robot considerably. The only downside to this approach is

that the robot began to suffer from the R2D2 affect, which is when a robot begins to look like a mobile garbage can or appliance. The improved layout of hardware is shown in Figure 2-1.

2.2 Sensors

The sensors used for the robot needed to be able to quickly and accurately detect obstacles in the robot environment, to allow for reasonable odometry across distances of 10-20m, and to allow for tracking ceiling-mounted fiducials.

For obstacle avoidance, two SICK LMS291 laser rangefinders are used. These devices provide an accuracy of $\pm$0.05m for distances of 30-40m and scan at 75Hz at an angular resolution of 0.0175 radians, thereby meeting both requirements for obstacle detection. One lidar is mounted parallel to the ground at a height of 0.64m. This lidar provides the primary data for detecting obstacles around the robot and for building a map of the environment. The other lidar is mounted at a height of 1.24m and is inclined at an angle of 0.45 radians. The beams from the angled lidar hit the ground 2.57m in front of the robot. The primary purpose of this lidar is to find negative (stairs) or positive (small animals) objects that are outside the plane scanned by the lidar mounted parallel to the floor. A downside to the use of lidar as the primary sensor is the propagation of glass-walls around the MIT campus. Lidar cannot see these walls, which poses a difficult problem for safe navigation.

Quadrature encoders from USDigital and a Microstrain 3DM-GX1 inertial measurement unit (IMU) are used for odometry. The encoders provide 2000 counts per revolution, giving a theoretical distance resolution of 0.0004m, though this value is much less in practice due to wheel slippage. The IMU has an accuracy of $\pm$0.035 radians. The intention was to use the IMU for calculating orientation and the encoders for calculating distance. However, I was unable to ever get reliable data from the IMU. Several different mounting positions were tried and in all cases the proper calibration procedures were run. Regardless though, once the robot had driven for two to three minutes, the filter used by the IMU would get lost and report that the

robot was turning rapidly even while it was driving in a straight line or sitting idle. While the raw data from the sensors on the IMU was available, I found that odometry from the encoders alone was accurate to meet the desired 10-20m range, and the IMU was abandoned.

Global orientation of the robot is provided by ceiling-mounted fiducials. To read these fiducials, a camera facing the ceiling is mounted at the top of the 80/20 spine. A camera with interchangeable lenses and a small profile was needed. The chosen camera was a Prosilica EC750C. The camera uses a IEEE 1394A interface and provides a resolution of 720x480. In black-and-white, the camera can capture up to 60 frames per second (fps), and in color, up to 20 fps. The camera uses CS-mount lenses, and the current lens has a focal length of 2.8mm.

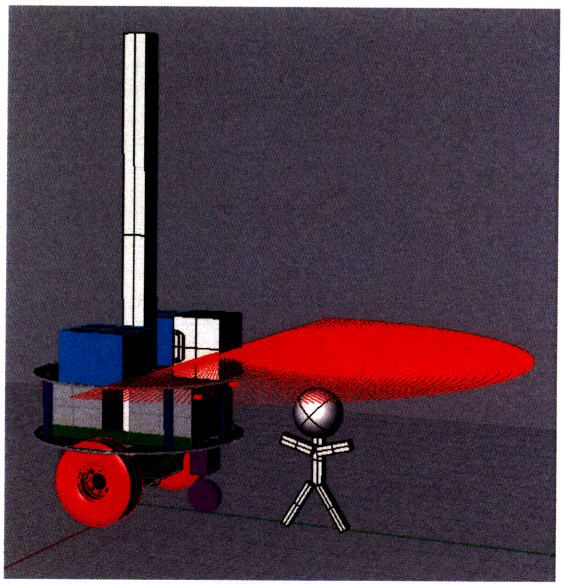


Figure 2-2: An early rendering illustrating the need for the angled lidar

2.3 Computation and Display

All computation for the robot is handled by an on-board computer. The computer, manufactured by Global American Inc., is a Mini-ITX motherboard with an Intel Core2 Duo T7200 CPU running at 2.0GHz. The computer has an 8GB solid-state hard drive and 2GB of DDR2 RAM clocked at 667Mhz. The computer runs the OpenSUSE distribution of the Linux operating system. A general computer, as opposed to a microcontroller- or FPGA-based, system was chosen because it was flexible, easy to get running, and provides for easier development cycles.

As a tour-guide robot, Tourbot needed interactive and multimedia features. These are accomplished using two LCD screens. One LCD screen is mounted at a sharp angle and has touchscreen-capability. This screen is the primary interaction device, allowing users to select tours and find information. The other LCD screen is mounted vertically and is used for media playback. As the robot leads, it plays various forms of media that are displayed on the vertical screen.

2.4 Electrical System

The devices on the robot run at 24V, 12V, and 5V. The layout of the system can be seen in Figure 2-3. It was decided to run the main power for the robot at 24V because the most power-hungry devices, the robot's motors, operate at 24V. To supply the power, the robot uses two 12V 100 Amp-hour lead-acid batteries wired in series. The 24V are fed into a 24V-12V DC/DC converter which outputs the 12V needed to run the robot's audio, computer, and displays. Servos used on the robot require 4.6-6.0V. The 12V from the converter is fed through a 7805 5V regulator to provide power.

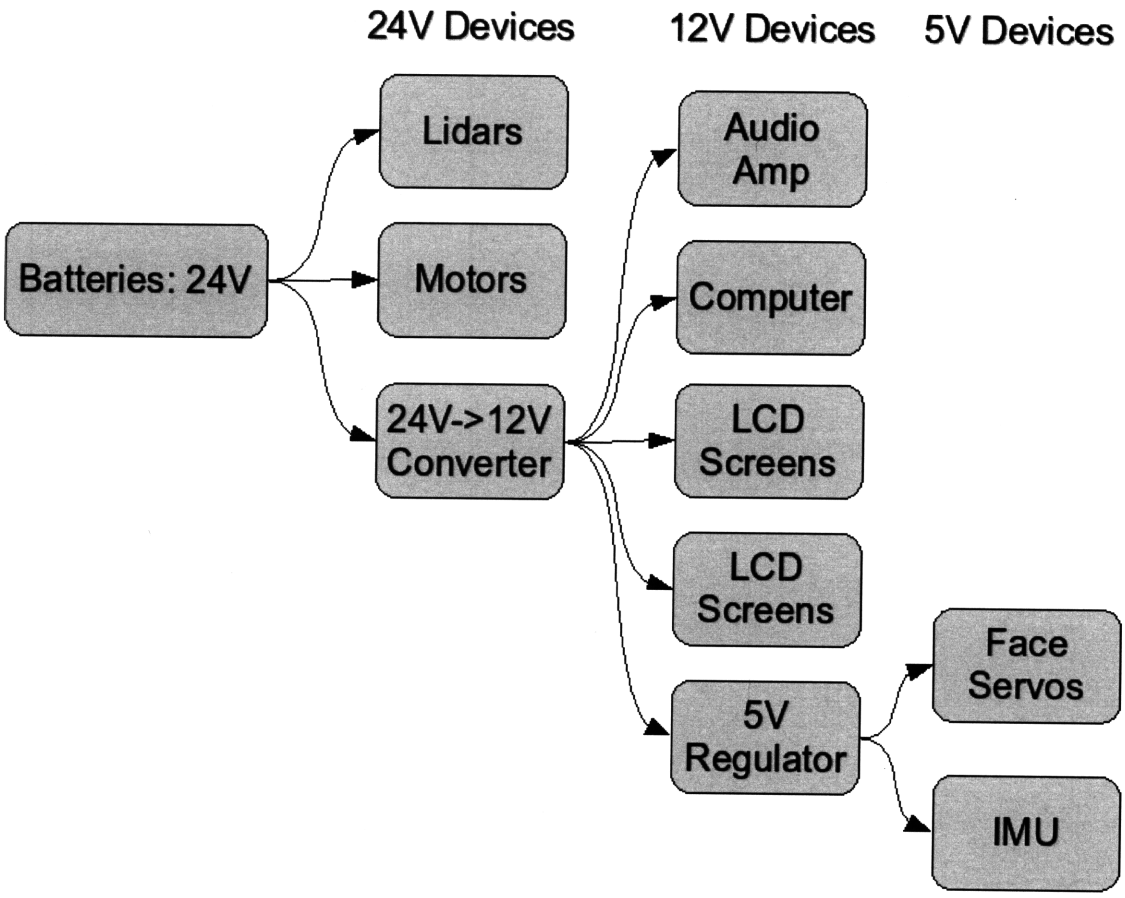


Figure 2-3: Electrical Connections for Tourbot

Chapter 3

Software: The Brains

This chapter describes the software used by Alan Touring to navigate his world and interact with tourists. The software for the robot followed the same principle as the hardware: you learn by doing. Consequently, all software was written from scratch using the C++ programming language. The only libraries used were low-level drivers or libraries that provided a direct hardware abstraction layer, like libdc1394, or service outside the scope of the project, like text-to-speech capability. At the time of this writing, the codebase for the robot consists of over 140,000 lines of code including documentation, though the code still in use is less than this number. I developed 100% of the software currently running on the robot and about 98% of the total software over the life of the project.

Writing all the code from scratch increased the overall development time, but again provided the benefit of learning a great deal about every aspect of software development for robots. In addition to the algorithmic side of robot software, I dealt with the intricacies of interfacing with sensors, efficiently handling inter-process communication, writing useful data visualizations, and dealing with keeping a complex system functioning.

3.1 System Design

The design for the software system is one of specialization and abstraction. Each functional piece runs in a separate process, and data is transferred between processes using a message-based communication protocol. Separating the different parts of the robot functionality into separate processes simplifies the overall system by providing a strong layer of abstraction and by isolating programming errors. If a process needs a particular type of data, there is a single means of accessing it via a well-defined interface. If a process crashes, then certain messages will no longer be produced, but the system as a whole remains healthy.

The system is laid out in a hierarchy of functionality. At the lowest level of the hierarchy are the hardware interfaces that abstract away communications with the sensors and controllers used by the robot. These modules either forward information read from sensors to the rest of the system or translate commands from other modules into the communication protocol for the device. The next level of the hierarchy are the modules that process raw sensor data into more abstract constructs used by the next level in the hierarchy. For example, fiducials are extracted from the raw images and are provided to localization and path-planning modules. At the highest level of the hierarchy are modules that handle tasks like path-planning, localization, and interaction with humans. The overall hierachy of modules is shown in Figure 3-1.

3.2 Communication System

The communication system for the robot is an anonymous publish/subscribe architecture that uses TCP/IP sockets for data transfer. Each process in the robot's computer network subscribes to some messages and provides others. Direct TCP/IP connections are created between each provider and subscriber, rather than connecting to a single process that handles all message routing, to avoid the bandwidth bottleneck and a single point of failure of all processes on the computer. An example of how the communication system works is shown in Figure 3-2.

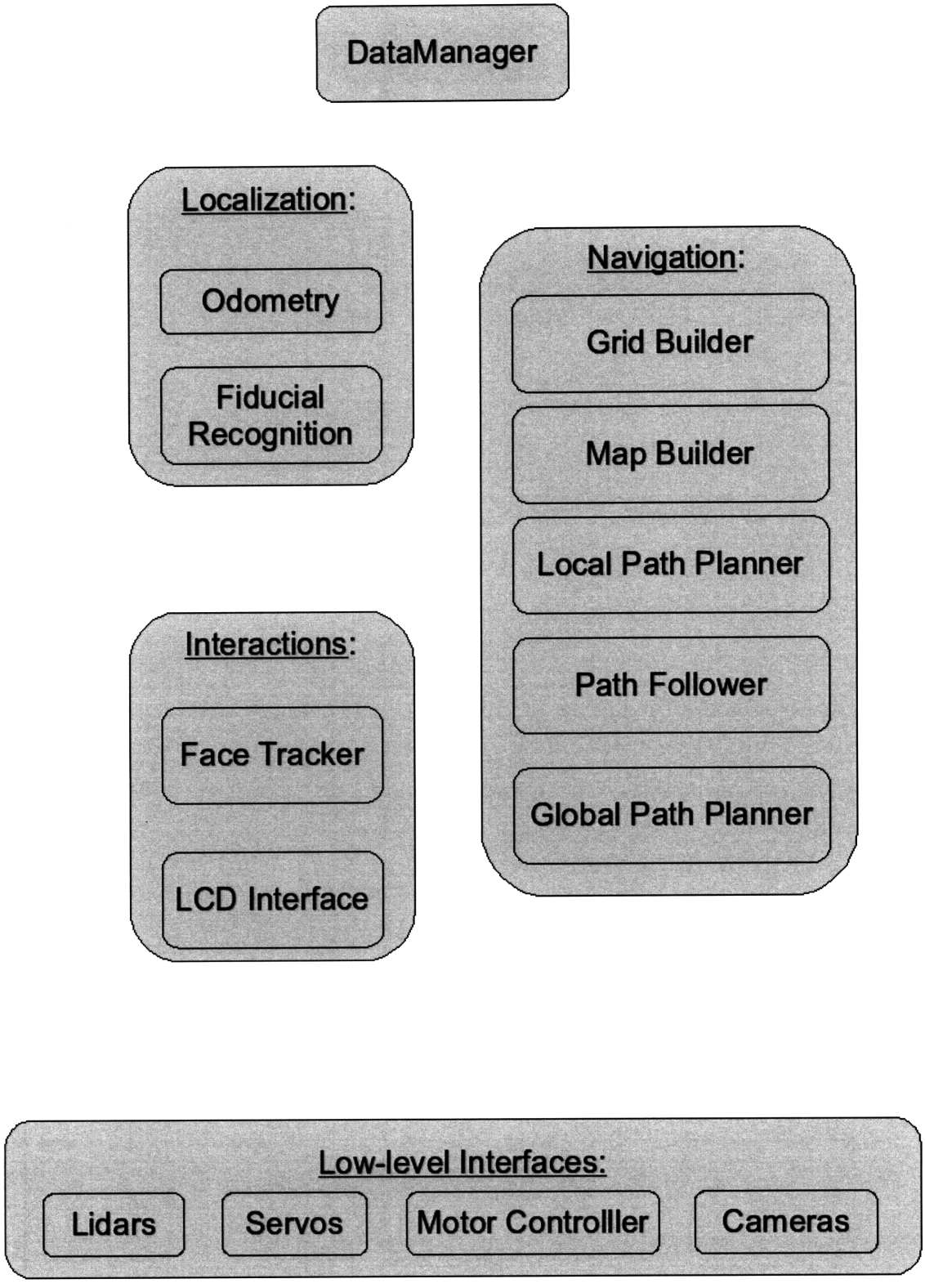
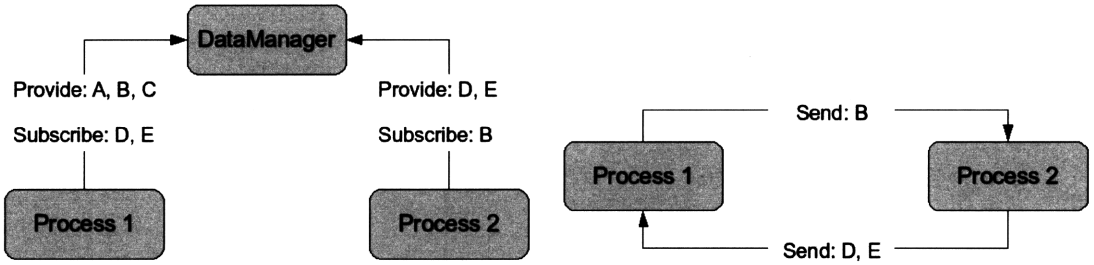


Figure 3-1: Organization of the software system



(a) Establishing subscriber/provider for processes

(b) Direct communication between processes

Figure 3-2: An Example of the Tourbot Communication System

3.2.1 DataManager

A process, the DataManager, runs on each computer used by the robot. The job of the DataManager is to link message subscribers and message providers. When subscribers or providers are received, the information is forwarded to all other active managers in the system. Thus, each DataManager contains all subscribers and providers on the whole network. When a new manager joins the network, it synchronizes its list of subscribers and providers with all other active managers. Doing this allows new computers to be dynamically added and subtracted from the system. The downside to the current manager system is that any new manager must know the host information (IP address and port) for at least one active manager. At the current time, a configuration file loaded by each manager on startup specifies the location of one or more computers in the network. A solution to this problem is addressed at the end of this section.

When a new message provider is added to the system, the manager checks to see if there are any subscribers to that message. If there are subscribers, the manager informs the new provider of the other processes' host information. Similarly, when a new message subscriber is added, the manager informs the subscriber of all providers for the message type.

3.2.2 DataMinion

When processes start, they connect to the manager and send messages indicating the types of messages that they provide and that they wish to receive. All communications for the processes are handled by an instance of the DataMinion created by each process on startup. The DataMinion establishes a connection to the DataManager on creation and subsequently supplies information on the provided and subscribed messages for the process. The minion provides both a polling and a callback-based interface for receiving messages from the system.

In polled mode, the minion places subscribed messages into a queue as they arrive. The size of the message queue is configurable. The messages can then be retrieved one at a time using the getMessage() method. If no message of the desired type is currently available, then no message is returned. The getMessage() method does not block. The polled interface is most useful for classes that need to update at a fixed frequency, like a PD controller for driving the robot, but do not have data arriving in fixed intervals. The polled approach also avoids the complexity of multi-threading and asynchronous arrival of messages.

The callback mode is used more frequently in the robot's software. In callback mode, each time a message is received a callback, handleMessage(), is issued to all handlers subscribed to the given message type. The handleMessage() method provides a handler with the newly received message. Each message type received by the minion operates on a separate thread and callbacks are issued as messages arrive from the network.

3.3 Low-level Interfaces

The Tourbot uses a wide variety of hardware devices, each of which needs an interface to communicate with the robot's computers. Most of the devices communicate using a standard RS-232 serial port. The hardware interfaces aim to provide a simple and (mostly) consistent way to access the hardware without worrying about the low-level details of communication with the device. Each interface receives messages that

are then converted into commands that the device understands, or sends messages containing the raw or interpreted data output by the device.

There are two basic means of communicating with each piece of hardware: polled and continuous. In general, sensors use a continous mode because the sensor is constantly producing data, which needs to be made immediately available to the rest of the system. Other hardware devices, like the motor and servo controllers, use a polled interface to retrieve data or send commands only when needed.

For polled hardware, the interface acts only as a simple API for communicating with the device. The interface is a wrapper class that deals with the low-level communication details and checksums. Communication with the device occurs only when one of the public methods for the interface class is called.

For continuous hardware, the interface serves two purposes. The first purpose is to provide a simple API for communicating with the device, and the second purpose is to handle the volumes of data being read by the device. The wrapper deals with the low-level communications and checksums like a polled interface, but it also needs to continually read data from the sensor so that it is available when another process needs it. To handle the task of reading all the data, a continuous interface spawns a background thread that continually reads the data provided by the device. A separate thread handles the task of transmitting the data to the rest of the system. This thread is synchronized with the thread that reads the data using a condition variable. This synchronization guarantees that the data provided is the most recent data available.

3.4 Localization

The robot localizes in the world using dead reckoning and a map of ceiling-mounted fiducials. The fiducials all have the same global orientation, allowing the robot to use the fiducials to update its global orientation. The fiducials do not, however, encode a global position. The map of fiducials is topological and contains the relative position of the fiducial nodes. Each node in the map contains a list of adjacent nodes and their relative positions. Using a relative map is beneficial because the map does not

need to be incredibly accurate. If the provided distances are close enough to the actual distances for the robot to have the fiducial in the field of view of the camera on arrival, then the map is functional and useful. The fiducials are placed five to ten meters apart, depending on the topology of the environment.

3.4.1 Odometry

Odometry for the robot is based purely on wheel encoders. Fiducials are placed close enough that this dead reckoning approach is sufficent to successfully navigate between them. The equations for determining the robot position from encoder values are:

$$\Delta\theta = \frac{(\Delta r - \Delta l)}{wheelbase} \tag{3.1}$$

$$\Delta d = \frac{(\Delta r + \Delta l)}{2} \tag{3.2}$$

$$x_{robot} = x_{robot} + \Delta d \cos(\theta_{robot} + \frac{\Delta\theta}{2}) \tag{3.3}$$

$$y_{robot} = y_{robot} + \Delta d \sin(\theta_{robot} + \frac{\Delta\theta}{2}) \tag{3.4}$$

$$\theta_{robot} = \theta_{robot} + \Delta\theta \tag{3.5}$$

3.4.2 Fiducial Recognition

Fast and accurate fiducial recognition is essential to the success of the robot. The processing needs to be fast to minimize the delay between when a frame is captured and when the fiducial processing has finished. A shorter time gap makes the global orientation updates more accurate because the robot will be closer to where the frame was captured. Additionally, faster processing allows for a higher framerate, which provides more opportunities to successfully locate a fiducial that enters the visual field of the ceiling-aimed camera. Accurate recognition is essential to ensure that the robot does not become lost while giving tours. False positives cause the robot to believe it is in a different part of the world which will cause the tour to fail and the wrong media to be played. False negatives can also cause the robot to become lost,

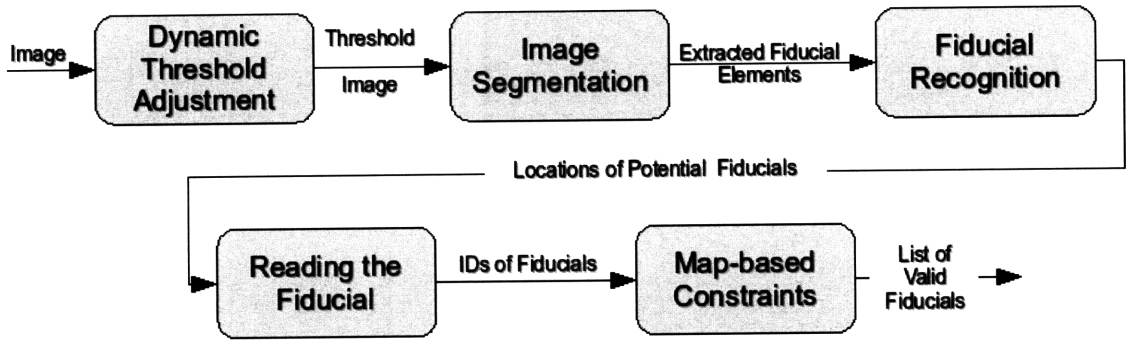


Figure 3-4: Flow chart of data through the fiducial recognition system

but the effects are not guaranteed to be as deleterious as a false positive.

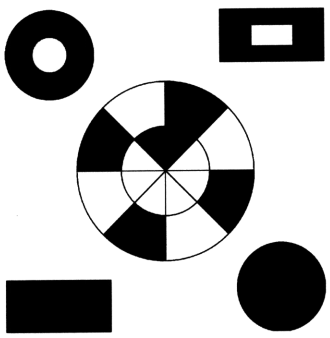


Figure 3-3: A fiducial used by the robot

The fiducial recognition assumes that the ceiling is of uniform color and that the color is the opposite color of the fiducial. On a black ceiling, the fiducial components are white, and vice versa for a white ceiling. This assumption allows fiducials to be found extremely fast. The algorithm for locating fiducials in an image has five steps (see Figure 3-4). First, the white/black threshold is adjusted based on the brightness values in the image. Next, the image is segmented using a simple binary test. Third, the raw segments are passed through a series of constraints to see if some combination of the segments creates the correct shape for a fiducial. Any constructed fiducials are then read. The final step checks the ID of the fiducial against the current map and filters them based on Hamming distance.

Dynamic Threshold Adjustment

Fiducials are mounted on the ceilings because the ceiling is a mostly static environment. The main element that changes on the ceiling is the lighting. Depending on the time of day or location, the illumination on the ceiling can vary drastically. The different lighting conditions wreak havoc on the image segmentation. Consequently,

the boundary between black and white pixels needs to be constantly updated based on the immediate lighting conditions in order to successfully find the fiducials in an image. The threshold adjustment is based on the previously stated assumption that the ceiling is a single color. If the ceiling is uniformly colored, then most of the pixels in the image will correspond to a single brightness value, and all pixels that don't correspond belong to the fiducial. This condition does not actually hold in the real world, but it is the basis for the thresholding algorithm described below.

The threshold adjustment algorithm tries to identify the two biggest brightness peaks in a histogram of pixel brightnesses in the image. These peaks typically correspond to the dominant ceiling color and fiducial color. The histogram contains 128 bins and is constructed by sweeping through each pixel in the image and incrementing the bin corresponding to brightness / 2. Different bin sizes were tested and 128 bins worked the most reliably.

Once the histogram is created, the largest negative derivative between two adjacent bins is found using $\delta_{brightness} = (bins[x-1]-bins[x])/bins[x]$ for the derivative calculation. The largest negative derivative is used because it indicates the tail end of the highest peak. Now that the highest peak has been identified, the second highest peak can be found by searching for the highest peak that lies to the right of the previously found peak. In this case, the derivative is $\delta_{brightness} = (bins[x+1] - bins[x])/bins[x]$ because you want to find the rising edge of the peak so that the threshold includes most of the fiducial pixels. This method works for finding the threshold when the ceiling is black, when the ceiling is white, then the search for the peaks should go in the opposite direction.

Image Segmentation

Image segmentation separates the individual fiducial elements from the ceiling. Each element is identified according to its shape and brightness as being one of the four unique corners of a fiducial: circle, circle with hole, rectangle or rectangle with hole (Figure 3-3). This process has two phases. First, all elements not belonging to the ceiling are found. Then the elements are categorized. The following description is for

white fiducials on a black ceiling.

To find the fiducial elements, a binary segmentation is performed using the threshold discovered in the previous stage of the image processing. A scan of the image pixels begins in the upper left corner of the image to extract all potential fiducial elements from the image. During the scan, each pixel is checked to see if it is above the black threshold. If a pixel is found to be above the threshold, then a breadth-first search using 8-way connectivity is run starting at the pixel location. During the search, all pixels found above the threshold are associated into a single region. The region is represented by the following values: $\sum x$, $\sum y$, $\sum x^2$, $\sum y^2$, $\sum xy$, and the total number of pixels in the region.

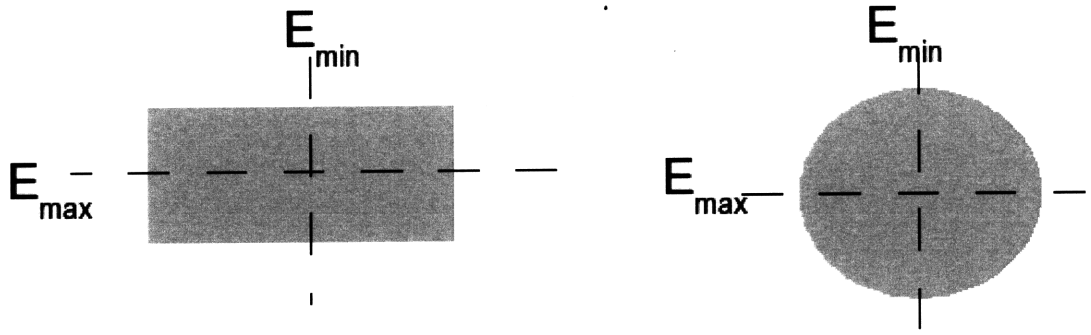


Figure 3-5: Comparsion of E_{min} and E_{max} for the fiducial shapes

Once all the regions in the image have been extracted, they are classified using a two-step process. The first step looks at the pixels in the center of the region. If they are black, then the region is either a circle with a hole or a rectangle with a hole, otherwise it is a circle or rectangle. The second step of the classification determines the shape of the region based on its roundness. Roundness is measured on a scale of 0 (a line) to 1 (a circle). Regions above a certain roundness value are classified as circles, otherwise they are rectangles. The roundness is found using the first and second moments of the image as calculated during the extraction [10]:

$$a = \sum x^2 - \frac{(\sum x)^2}{area} \tag{3.6}$$

$$b = \sum xy - \frac{\sum x \sum y}{area} \tag{3.7}$$

$$c = \sum y^2 - \frac{(\sum y)^2}{area} \tag{3.8}$$

$$\sin(2\theta) = \pm\frac{b}{\sqrt{b^2 + (a-c)^2}} \tag{3.9}$$

$$\cos(2\theta) = \pm\frac{a-c}{\sqrt{b^2 + (a-c)^2}} \tag{3.10}$$

$$E_{min} = \frac{1}{2}((a+c) - (a-c)\cos(2\theta) - b\sin(2\theta)) \tag{3.11}$$

$$E_{max} = \frac{1}{2}((a+c) + (a-c)\cos(2\theta) + b\sin(2\theta)) \tag{3.12}$$

$$roundness = \frac{E_{min}}{E_{max}} \tag{3.13}$$

Regions found during image segmentation are also filtered based on their areas. If the area of a region is outside of the $[a_{min}, a_{max}]$, then it is thrown out. This filter is used to get rid of very small segments that are the result of noise and very large segments that are the result of a poor threshold. The range of valid areas varies based on the height of the ceiling to which the fiducials are mounted.

Fiducial Recognition

Fiducials are found from the extracted elements by grouping the elements based on type and the passing them through a series of constraints intended to filter false positives. There are four constraints, each designed to determine how far the potential fiducial varies from the model fiducial. When the raw elements are received from the image segmentation, they are split into four groups based on their classification. After being split, all possible combinations are checked to see if they create a valid fiducial. This operation is potentially expensive if many elements of each type are segmented from the image because it grows geometrically. However, in practice, the segments segmented from the image that are not actually part of a fiducial tend to all be classified as rectangles because the requirements for a circle are unlikely to happen as the result of random noise.

The first constraint applied to the potential fiducial checks to see that angles formed by the corners of the fiducial fall within some tolerance of the actual angles.

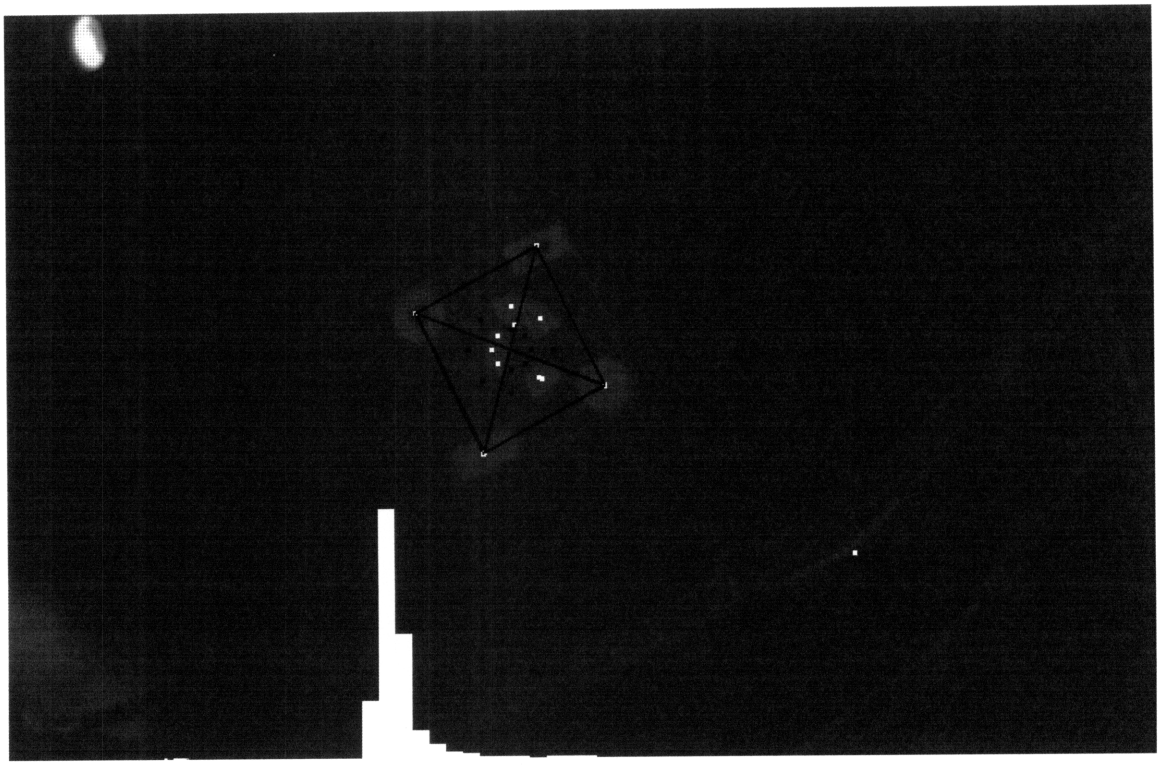


Figure 3-6: A fiducial extracted from an image in the MIT Museum. Across the bottom of the image is the histogram of brightness values used for the threshold adjustment. Notice the very high peak corresponding to the ceiling color.

For the fiducial design that is used, the angles formed with rectangles at the vertex are 1.52 radians, and the angles formed with circles at the vertices are 1.62 radians. After checking the angles, the lengths between the fiducial corners are checked to see that they are all approximately the same. The areas are also compared to see if they are the same. The final constraint ensures that the layout of the fiducials is a rectangle by checking the angles formed between the center and adjacent corners satisfy the correct ordering of elements around the border of the fiducial.

Once the fiducials have been found, the orientation of the fiducial in the image is calculated. To find the orientation, the average angular displacement of the corners is used. The angular displacement is how far the angle vector pointing from the center of the fiducial to the given corner deviates from the angle of the vector between the center and the corner when the fiducial is oriented to 0 radians.

Reading the Fiducial

The 16-bit identifier for the fiducial is encoded in two concentric circles in the fiducial's center. The circles are split into eight regions and colored either black or white. The radii of the circles are found in proportion to the side lengths of the extracted fiducial. The ratio of inner radius and outer radius to the side length is provided as a parameter in a configuration file. The side length used for finding the radii is the average side length of the fiducial. After the side length is found, the fiducial is read by reading counterclockwise around the fiducial, starting with the outer circle and then the inner circle. The angle that each section is located at is $\theta_i = \theta_{fid} + \frac{\pi}{4}i + \frac{\pi}{8}$. The additional $\frac{\pi}{8}$ is the offset to the center of each region. The value of each section is read as the median color of a 5x5 square of pixels centered at $(r_i \cos(\theta_i), r_i \sin(\theta_i))$.

Map-based Constraints

The above steps successfully read the fiducials that are located in the image, but the results are not 100% accurate. The biggest problem is that noise can cause the estimated the centers of the fiducial corners to be offset by a few pixels. For the calculation of the fiducial orientation, it is not a problem because the deviation is not large enough to cause a major angle misreading and the effect of noise averages across frames. But for reading the fiducial identifier, the noise can throw the reading off by a number of bits.

To mitigate the misreading of a fiducial, the last step in the extraction algorithm compares the identifier that was read against the identifiers for all fiducials in the map. A match is made if the Hamming distance between the read fiducial and a map fiducial is one or less. This final filter ensures that false positives are extremely rare.

3.5 Navigation

To give tours, the robot needs to be able to navigate through the world. The navigation process is divided into three steps: mapping, path planning, and path following. The flow of data is shown in Figure 3-7.

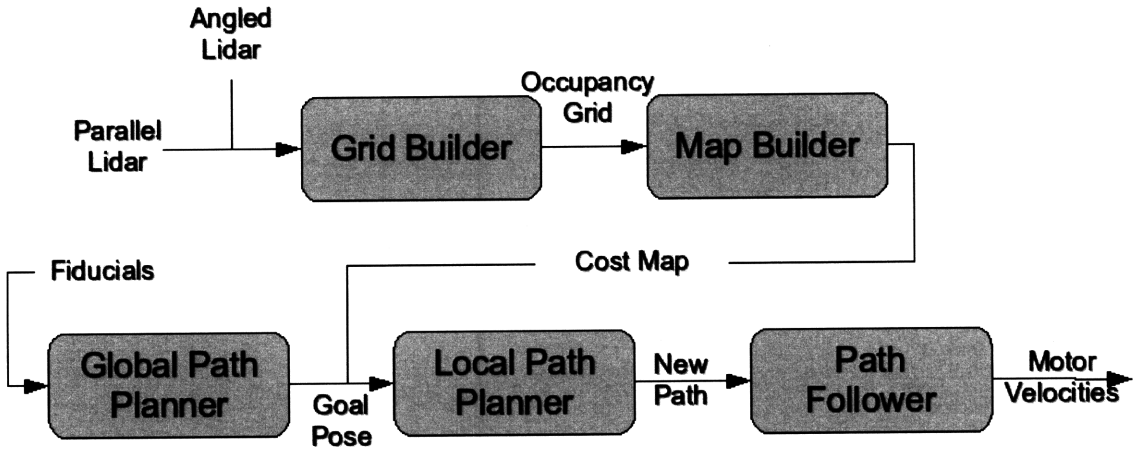


Figure 3-7: Flow of data through the navigation system

3.5.1 Mapping

The only pre-built map the robot is provided contains the relative location of the fiducials in the world. No information about the physical world is given. Consequently, the robot must build a map of its surrounding in real-time. The map is built using raw lidar data. The map is then converted into a cost map in configuration space, which is then used for path planning.

Occupancy Grid

The map of the robot's surrounding is built using the occupancy grid algorithm [29]. The occupancy grid represents the world as a two-dimensional grid of evenly-spaced cells. On each update of the map, the cells in the grid are checked against the sensor values. Any cell that was perceived by a sensor has its estimate updated. The algorithm itself is quite simple, but the accuracy depends heavily on the sensor model that is used for the updates. Furthermore, the method for determining if a cell was seen affects the efficiency of the map updates. An additional problem that had to be solved with my occupancy grid implementation was the use of lidars with different perceptual fields.

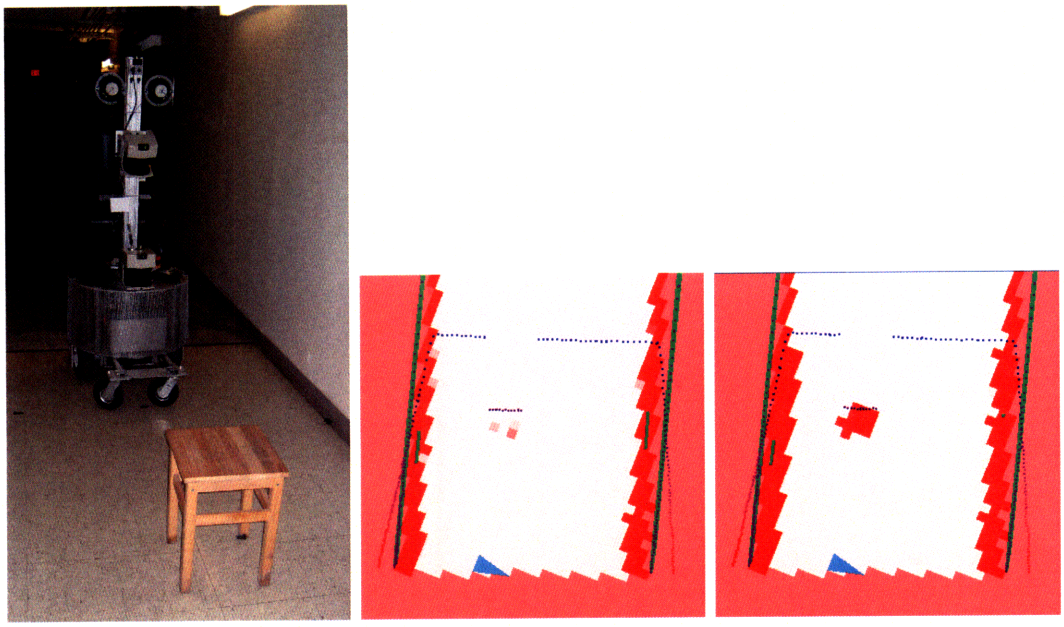
Each cell is a binary random variable that estimates the likelihood that the cell is occupied. The probability estimate for a cell is $p(\mathbf{m}_i \mid z_{1:t}, x_{1:t})$, where $\mathbf{m}_i$ is

the likelihood the cell is occupied, $z_{1:t}$ are the sensor measurements and $x_{1:t}$ are the robot positions. This estimation problem can be solved with a binary Bayes filter using a log-odds likelihood representation. The binary Bayes filter is a recursive state estimation method, simplifying the problem by making the current estimate reliant only on the previous estimate rather than all previous measurements and positions. The update equation for a cell then becomes $o_{i,t} = o_{i,t-1} + likelihood_t - o_0$, where $o_{i,t} = \log \frac{p(\mathbf{m}_i|z_{1:t},x_{1:t})}{1-p(\mathbf{m}_i|z_{1:t},x_{1:t})}$, the log-odds likelihood that a given cell is occupied and $likelihood_t$ is the likelihood that the cell is occupied given the current sensor measurements.

The sensor model used relies on two values of the lidar sensor, its divergence, ω and initial beam width, λ. With these values, the cone of space that the beam occupies can be calculated. The cone is then discretized using the same cell size as the occupancy grid. To update the map when a new measurement is made, this discretized cone is translated and rotated to the correct position on the grid. The cells to update are then easily read. The likelihood value of a cell is based on sensor reading. All cells closer than the read distance are considered "free" and decrease the likelihood of occupancy. The cells that are at the read distance are marked as occupied and the likelihood of occupancy increases. All cells further than the read distance are ignored. This method of performing the updates works well because only the cells that are affected by the new measurements are updated while the rest are ignored, thereby decreasing the computation time of an update.

The above model is the traditional approach to occupancy grids and works very will with a single lidar. However, the robot has two lidars oriented at different angles. The lidar mounted parallel to the floor scans a plane parallel to the floor, while the angled lidar scans a plane that intersects the floor. The problem this creates is that a traditional 2D occupancy grid cannot correctly represent the sensor data being collected by the robot. Readings from one lidar will erase the results of the other and vice versa.

Imagine a stool 0.5m tall. The parallel lidar will never see this stool because it only scans the plane 0.64m above the ground. The angled lidar will see this stool



(a) Invisible Stool

(b) Original Occupancy Grid

(c) Improved Occupancy Grid

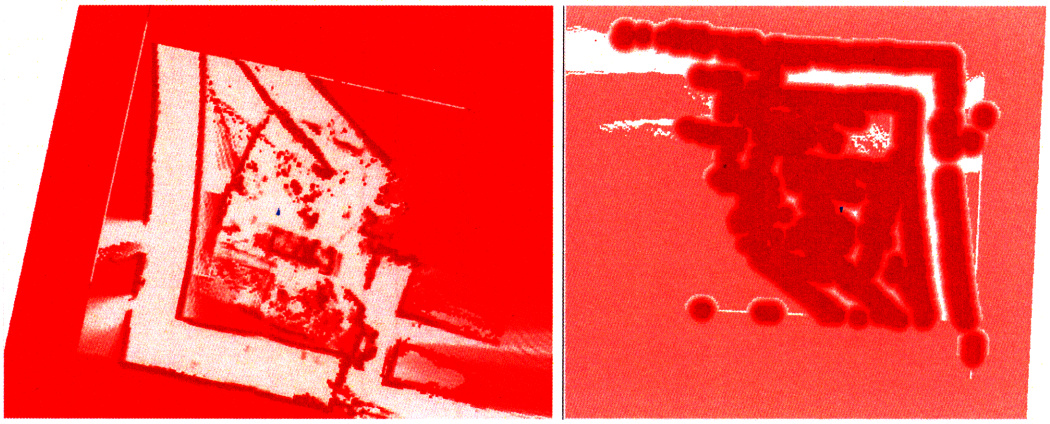
Figure 3-8: A stool that is invisible with the original 2D grid. Notice the much greater certainty that an object is in the path of the robot with the improved occupancy grid versus the original 2D grid.

only if it is in front of the robot and within 2.57m. When the angled lidar does see the stool, the likelihood that the cell is occupied increases as would be expected. However, the next scan taken by the parallel lidar will fail to see the stool and will report the location as unoccupied and update the likelihood accordingly. The net effect is no change in the likelihood, making the stool effectively invisible despite it being in the field-of-view of the robot's sensors.

One potential solution to this problem would be to extend the occupancy grid into three dimensions. The height that a reading is made at is discretized along with the (x, y)-coordinates and stored in a 3D grid. However, the robot only plans in a 2D configuration space, so the third dimension needs to be projected into the plane. Additionally, the extension to the third dimension greatly increases the memory footprint of the occupancy grid, which will have a negative impact on performance.

An alternate solution for building the occupancy grid augments the 2D grid by storing the maximum height of an occupied reading for each cell. When an update occurs, one of four conditions arises. If a cell is read as occupied, then its occupied likelihood increases accordingly. In the case that the height is greater than the previous maximum, then the maximum occupied height is updated. If a cell is read as unoccupied and the height is less than the current maximum height, then the cell's occupied likelihood will be decreased as normal and the maximum occupied height will be set to the height of the current reading. The height change is made because the new scan indicates that there is nothing above it, but provides no information on whether there is anything occupying the space below the scan. Finally, if the cell is read and unoccupied and the height is greater than the maximum occupied height, no update of the likelihood occurs. No update occurs because the scan contains no information about the part of the cell that was previously found to be occupied.

This approach to building the occupancy grid partially solves the invisible stool problem while using much less memory than the full 3D occupancy grid. In particular, the invisible stool problem disappears for all objects less than or equal to the height of parallel lidar. An object with a maximum height less than the scan height of the lidar will remain in the map. The invisible stool problem remains, however, for objects like



(a) Occupancy Grid (b) Cost Map

Figure 3-9: An occupancy grid and cost map built by driving the robot around its lab. The curved hallway shows the robot's odometry error. Note that the vast majority of the world is considered untraversable due to the extra large configuration space.

tabletops that exist only above the plane scanned by the parallel lidar because their maximum height will be adjusted down to the height of the parallel lidar and updated as being unoccupied. Not seeing these taller objects is not acceptable though, as the angled lidar was added specifically to avoid the potential for the robot clotheslining itself on a table.

Ultimately, my implemented solution uses the maximum height-based occupancy grid with a modification to handle the invisible stool problem for taller objects. When the updates to the grid are happening, if the measured height is at the same height as the lidar, meaning it came from the parallel lidar, the cell will only be marked as unoccupied if the occupied height is either the same height, or zero, meaning completely unmeasured. This approach allows taller objects to be seen by the angled lidar and remain in the map, despite being seen as unoccupied by the parallel lidar. The map is 30m x 30m with a cell resolution of 0.1m.

Cost Map

The occupancy grid provides an accurate map of the robot's immediate surroundings, but is not satisfactory for planning. The map used for planning needs to be in the

configuration space of the robot. Additionally, the planning algorithm relies on the costs for different actions to find the best route. To handle these planning necessities, a cost map is constructed from the occupancy grid for use in planning.

The cost map converts the occupancy grid into the configuration space of the robot. To simplify the planning problem, a basic circular model of the robot is used. The robot is treated as a circle with a radius of 0.6m, which is the length from the center of the robot to the end of the front wheels. Using this configuration space adds 0.2m to the width of the robot, which makes some areas of the world seem untraversable when the robot could actually navigate through them. However, this configuration space makes the planning problem a search in a two-dimensional space where the generated path is a series of (x, y)-coordinates. A more appropriate representation of the robot would be a rectangle or hexagon, but using these shapes would push the planning problem into three dimensions, producing a path as (x, y, θ)-coordinates. The looser bound has not caused failures with the robot, making the tradeoff worthwhile due to the simpler planning task.

The occupancy grid represents the world as cells containing the likelihood that a given position is occupied in log-odds format. The cost map classifies each cell as unoccupied, unknown, or occupied based on the likelihood. If the grid is 95% sure that a position is unoccupied it is marked as such. Similarly, if a position has an occupied likelihood of over 95% it is marked as occupied. Otherwise, the region is considered to be unknown. The current implementation has a cost of 0 for free cells, 5 for unknown cells, and 71 for occupied cells.

When an occupied cell in the grid is found, it is converted to configuration space by overlaying a discretized circle representing the footprint of the robot onto the cost map. Each cell that contains a piece of the circle is assigned the cost of an occupied cell. The cost outside the circle falls off to zero across a configurable radius. This additional falloff keeps the robot from travelling close to obstacles unless no other route exists, and when two obstacles are too close, the generated path will be equidistant from the obstacles.

3.5.2 Path Planning

Path planning finds the path that the robot travels as it moves through the world. The path planning approach for the robot is two-tiered. A global planner determines the series of waypoints through which the robot moves while giving a tour, and a local planner handles the motion planning task for the robot.

Global Path Planning

When loaded, the global path planner receives a set of waypoints that, when followed in order, comprise a tour. The waypoints are not necessarily continuous, but rather they indicate the points of interest on the tour. The goal of the global path planner is to find sequence of fiducials within the relative map of fiducials to follow in order to successfully move through the world. The other task of the global planner is to determine when a waypoint has actually been reached by interpreting the output of the fiducial recognition system.

The relative fiducial map forms an undirected graph where the edges are all adjacent fiducials and edge costs are the distances between the fiducials. Using this graph, the global planner uses an A* search to find the shortest route between tour waypoints. To build the overall path, the paths between the adjacent fiducials are concatenated into a single path that is then followed by the robot. As the fiducials in the path are encountered, the planner directs the robot to the next waypoint based on the relative position of the next waypoint and the current global position of the robot.

Determining when to command the next waypoint for the robot to move to is the other half of the global path planner's task. The fiducials only encode the orientation of the robot, so the planner only has access to the position of the fiducial in the coordinates of the image in which the fiducial was found. Therefore, the time at which the next waypoint is assigned will affect how likely it is that the robot will reach the next stop in the global path. The time that a fiducial is triggered uses two criterion. The first criterion compares the distance of the fiducial from the center

of frame in which it was captured. While the robot is moving towards the fiducial, the distance will be decreasing. Once the robot is moving away from the fiducial, then this distance will begin to grow. At this point, the robot has reached its closest approach to the fiducial and should now move to the next waypoint in the global path. It might be the case, though, that the robot only gets a cursory glance of the fiducial or that the robot is not moving when the fiducial comes into view. To ensure that the fiducial is triggered, a count is kept with the number of times that a fiducial is seen while the robot is stationary. If this count goes above a threshold, then the planner will also direct the robot to the next point in the global path.

The global path planner constantly monitors the path being followed and if a fiducial is triggered that is out of sequence of the current global path, then a new path will be found based on the current robot location. Constant monitoring and replanning helps to keep the robot from getting lost because it can be diverted from the desired position and as long as it eventually sees a fiducial then it can get back on track. Similarly, the robot can be started anywhere within its environment and can wander until a fiducial is located, at which point it can find its way to the starting location for a tour.

Local Path Planning

The local path planner tries to find a way for the robot to get from its current position to a desired goal position. The cost map built from the occupancy grid is used for finding the best path to the given goal location. After the initial path to the goal is found, the robot begins to follow the path. As the robot travels along the path, the cost map of the world will change based on movement of objects around the robot and the shifting field-of-view of the sensors. When a new map is received, the local planner replans the path from the current robot position. Due to this replanning strategy, an incremental planning algorithm, D*-Lite [14], is used for finding the robot's path.

The D*-Lite algorithm is an incremental algorithm which means that after the initial path is found, subsequent replanning takes advantage of the results of the previous search and only the modified parts of the map are considered. As shown

in [14], this approach results in a drastic decrease in the time needed for replanning. The algorithm begins the search at the goal and finds the best path to the starting position. All nodes expanded during the search contain the minimum cost needed to reach the goal from their position. The cost of a node is the cost of moving *to* the node *from* an adjacent node as determined by the cost map plus the minimum cost of adjacent nodes. In other words, $cost(i) = costmap_i + \min(cost(x \in Adj(i)) + dist_{x,i})$, where $dist_{x,i}$ is the physical distance from node x to node i.

D*-Lite is an informed search algorithm, so the choice of heuristic will have a noticeable impact on the overall performance of the algorithm. The heuristic used for the search is a slightly modified Manhattan distance. The normal Manhattan distance heuristic works for a four-way connected search graph, but my search implementation uses an eight-way connected graph. In an eight-way graph, the best possible path will move diagonally through the graph until the same row or column as the goal position is reached, at which point vertical or horizontal motion is used to reach the goal. Given this, the heuristic used for the search is:

$$x_{diff} = |x_{robot} - x_{goal}| \tag{3.14}$$

$$y_{diff} = |y_{robot} - y_{goal}| \tag{3.15}$$

$$h_{x,y} = \min(x_{diff}, y_{diff})\sqrt{2} + |x_{diff} - y_{diff}| \tag{3.16}$$

As can be seen, if the robot is in the same column or row as the goal, the heuristic is just the linear distance. If the the robot is directly diagonal, then the heuristic is diagonal distance. Otherwise, a combination of the two is used.

The performance of the planner is sensitive to the costs assigned to the different types of cells, and the occupied cost in particular. It is undesirable for the planner to plan paths through the occupied areas of the map, but setting the occupied cost too high results in extremely poor performance if the robot is on an occupied location. For example, a way to guarantee that an occupied cell will never be expanded is to set the cost of an occupied cell to be the same as the area of the map, in the case of the robot, 90000. Doing this, however means that if the robot is on an occupied cell, the

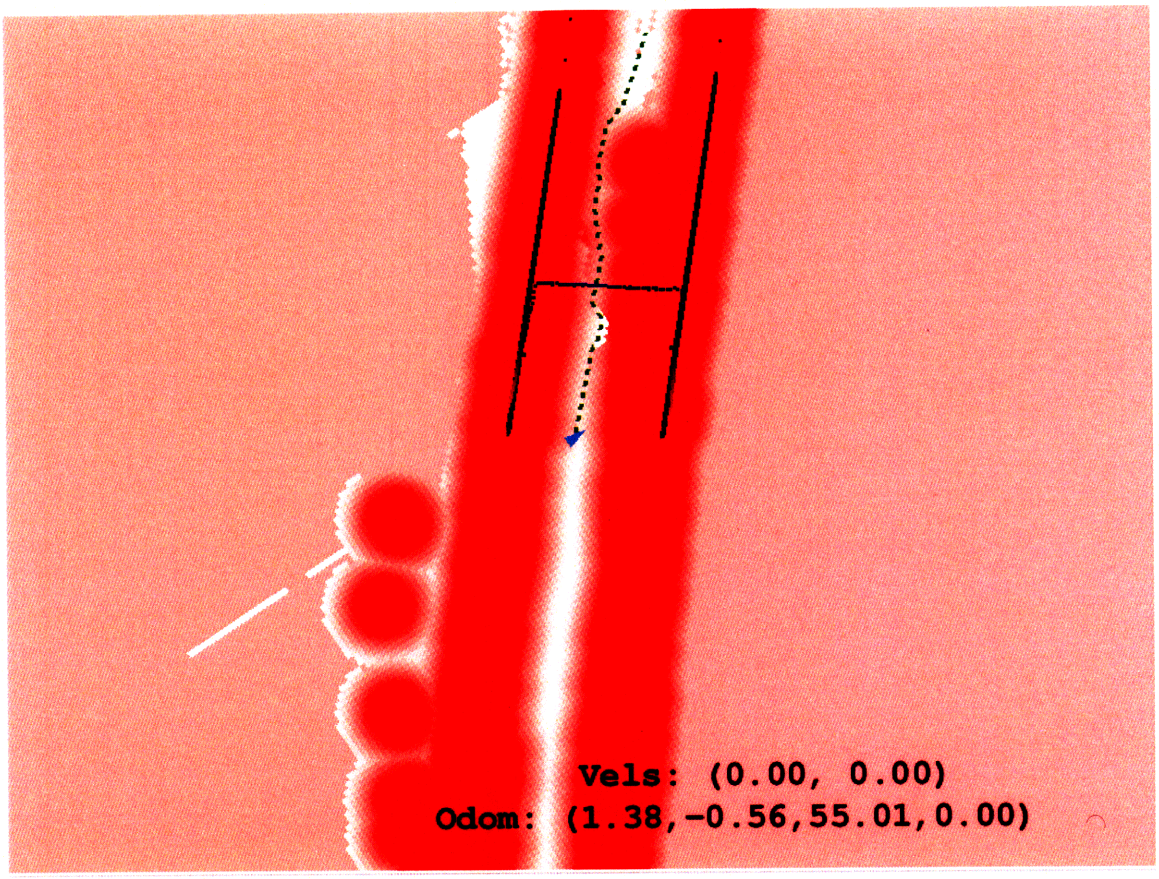


Figure 3-10: Path generated by the local planner through a cost map. Notice how the edge falloff in the cost map keeps the robot equidistance from obstacles.

search will expand every free and unknown node in the map before reaching the goal. Needless to say, this situation means that the initial planning takes a long time. But it is further aggravated by the fact that now every node has been expanded, meaning the search queue contains many nodes. When an update to the map occurs, updating the priority of the vertices that changed will be much more costly because insertions and deletions from the queue will be more expensive.

To assign the occupied cost of a search, I used a worst-case scenario analysis. The robot is in the top left corner; the map is split diagonally by a wall with a single opening in the bottom right corner; and the robot wishes to travel to the other side of the wall. The path needs to successfully go around the wall rather than through it. The total distance travelled is $600\sqrt{2}$ which is the distance to the bottom corner and back when moving along the diagonal. This distance establishes a lower bound

on the cost of moving through the wall. The thickness of the wall will be $\frac{d_{robot}}{l_{cell}}$. For the current robot configuration space, $d_{robot} = 1.2m$, and the map uses $l_{cell} = 0.1m$. Therefore, the width of the wall will be 12 cells. To ensure that the path goes around the wall $cost_{occupied} > \frac{600\sqrt{2}}{12} \approx 71$. Using this cost then minimizes the negative effects of the robot attempting to find a path from a position inside an occupied portion of the map, which can happen if the robot is surrounded by people.

Another problem with the straightforward approach of planning from the current robot position to the next goal occurs when the goal position is occupied. This scenario happens very frequently in the museum setting because the robot is moving between different displays in the exhibit, so people are often standing where the robot wishes to move. When the goal position is occupied, the least costly path is the path that moves to the closest unoccupied position and then moves around other occupied regions to the robot's position. Depending on the position of the goal within the occupied region, the path may not take the direct route to the goal but will instead meander around the edge of the map. If this happens, then the robot will become horribly confused because it will take off in a direction that carries it outside the fiducial map. It sometimes recovers if the alternate path keeps the actual goal in the field-of-view of the sensors and the obstacle at the goal moves, but it isn't guaranteed. The planner counters this problem by shifting the goal point if it is in occupied space. In particular, the goal is moved closer to the robot along the line connecting the current robot position and the goal until it is outside of occupied space and then the search proceeds as normal. The goal point is monitored the entire time and if it frees up, then the goal is shifted and a new path is found.

One additional trick that is used to help the robot successfully navigate is to pre-orient to the goal location before planning. When the planner receieves a new goal, it commands the robot to rotate and face the direction of the goal before planning. Doing this simple action helps the robot because it conveys the direction it will be travelling next to the people attending a tour, so they will move out of the way and also ensures that the robot is planning through the most up-to-date map possible.

3.5.3 Following the Path

Once a path has been found, the robot needs to follow it. The path follower is the most visible part of the robot's navigation system because it is the only conception people have of the robot's internal decision-making processes. If the robot's movements are very jerky, people are more apt to stay well away from it because the robot seems unpredictable and dangerous, whereas a smoothly moving robot projects its intentions much more clearly and seems more confident in its actions.

Initially, the robot would follow a path by driving from one vertex on the path to the next and reorient itself at each vertex. The motion that resulted from this path-following approach was very jerky and unnatural. If the robot was moving along the edge of a circle, it would appear to be moving in step-wise fashion due to the discretization in the grid-based plan. Some means of smoothing out the effects of the grid plan was needed.

The path following algorithm implemented on the robot smooths the motion of the robot by providing a constantly moving waypoint for the robot to reach. The translational and rotational velocities are then controlled based on the position of the waypoint relative to the robot. The general idea for adjusting the velocities is that the robot should travel at its maximum velocity when moving in a straight line and slow down as the rotational velocity increases. As the goal nears, the robot should gradually come to a stop. The easiest way to visualize the algorithm is to think of dangling a carrot in front of the robot and having him always drive towards the carrot. The algorithm for this is described below.

First, project the current position of the robot onto the path being followed:

$$ratio = \frac{((p_{next} - p_{start}) \cdot (p_{robot} - p_{start}))}{(p_{next} - p_{start}) \cdot (p_{next} - p_{start})} \tag{3.17}$$

$$p_{proj} = ratio * (p_{next} - p_{start}) + p_{start} \tag{3.18}$$

If $ratio > 1$, then the project exists past the end-point of the line, so increment to the next vertex in the path. Continue this until $ratio < 1$. Once p_{proj} is found, trace along the path from p_{proj} until $d_{carrot} = \lambda$ or the end of the path is reached, where

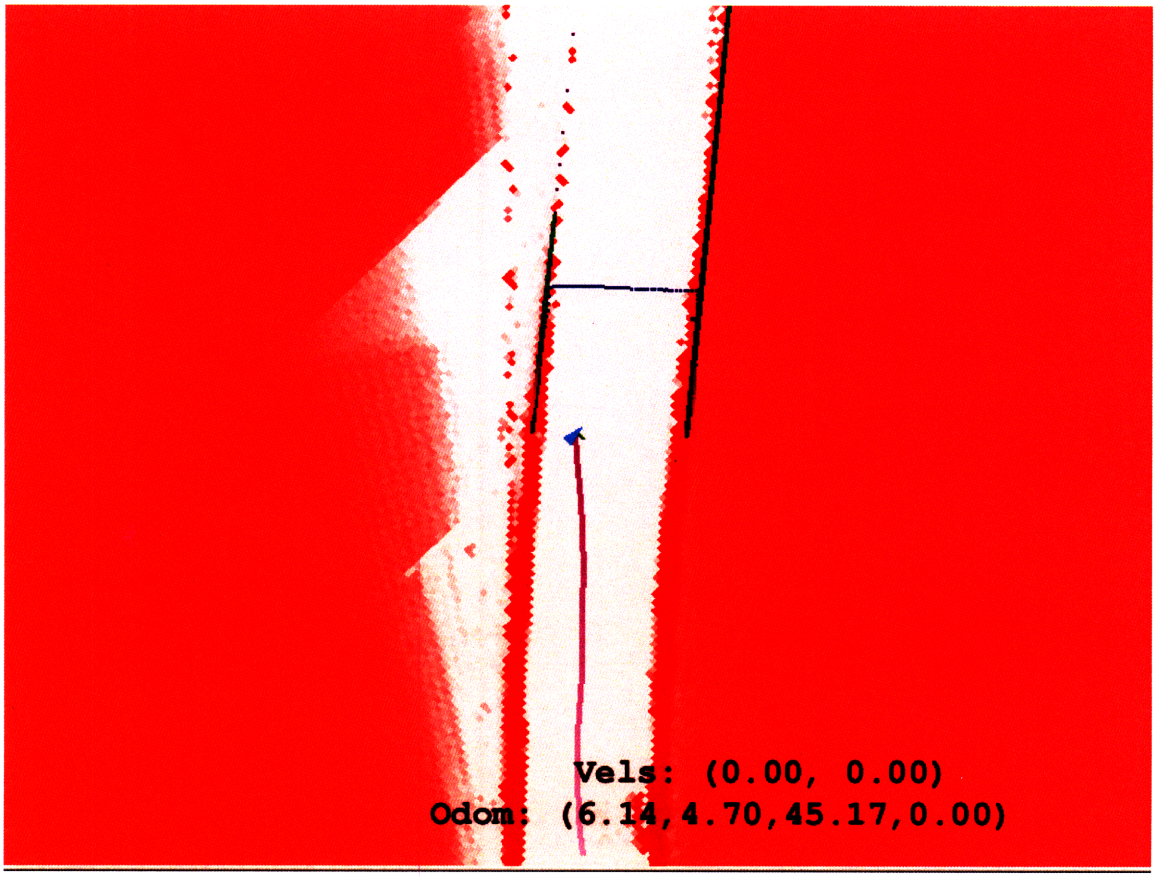


Figure 3-11: Trajectory followed by the robot along the path in Figure 3-10

d_{carrot} is the distance traced from p_{proj} and λ is the configurable distance to dangle the carrot in front of the robot. This point is p_carrot. The control law is based on the relation of p_{carrot} to p_{robot} as follows:

$$\theta_{diff} = \arctan(p_{carrot_y} - p_{robot_y}, p_{carrot_x} - p_{robot_x}) - \theta_{robot} \tag{3.19}$$

$$v_{desired} = v_{trans_{max}} \frac{d_{carrot}}{\lambda} \tag{3.20}$$

$$v_{trans} = \min(v_{desired}(1.0 - \frac{2|\theta_{diff}|}{\pi})\rho_{trans}, v_{desired}) \tag{3.21}$$

$$v_{rot} = \min(2\theta_{diff} v_{rot_{max}} \rho_r ot/\pi, v_{rot_{max}}) \tag{3.22}$$

where $v_{trans_{max}}$ is the maximum translational velocity, $v_{rot_{max}}$ is the maximum rotational velocity, ρ_{trans} is the gain for translation and ρ_{rot} is the gain for rotation.

3.6 Human Interface

A tour-guide robot is an interactive robot and as such needs a way to interact with people. The current interface has two parts: the face tracker and the LCD interface.

3.6.1 Face Tracker

The face tracker is extremely simple, but adds a great deal of expressiveness to the robot. The tracker uses the face finding algorithm available in the OpenCV library to find faces in the images being captured by one of the eye cameras. Once the faces in the image have been found, the face tracker chooses the largest face and commands the pan/tilt platform to aim towards the face. The tracker tries to orient the eye cameras to make the largest face lie at the center of the captured image. When a face is at the center of the image, the robot's attention will seem focused on that face.

Another benefit of this system is that the face finding algorithm will occasionally have false positives, which cause the eyes to orient themselves in that direction. This occasional glance to some random direction makes the robot seem curious and interested in its environment. These false positives do not arise often, so the eyes remain

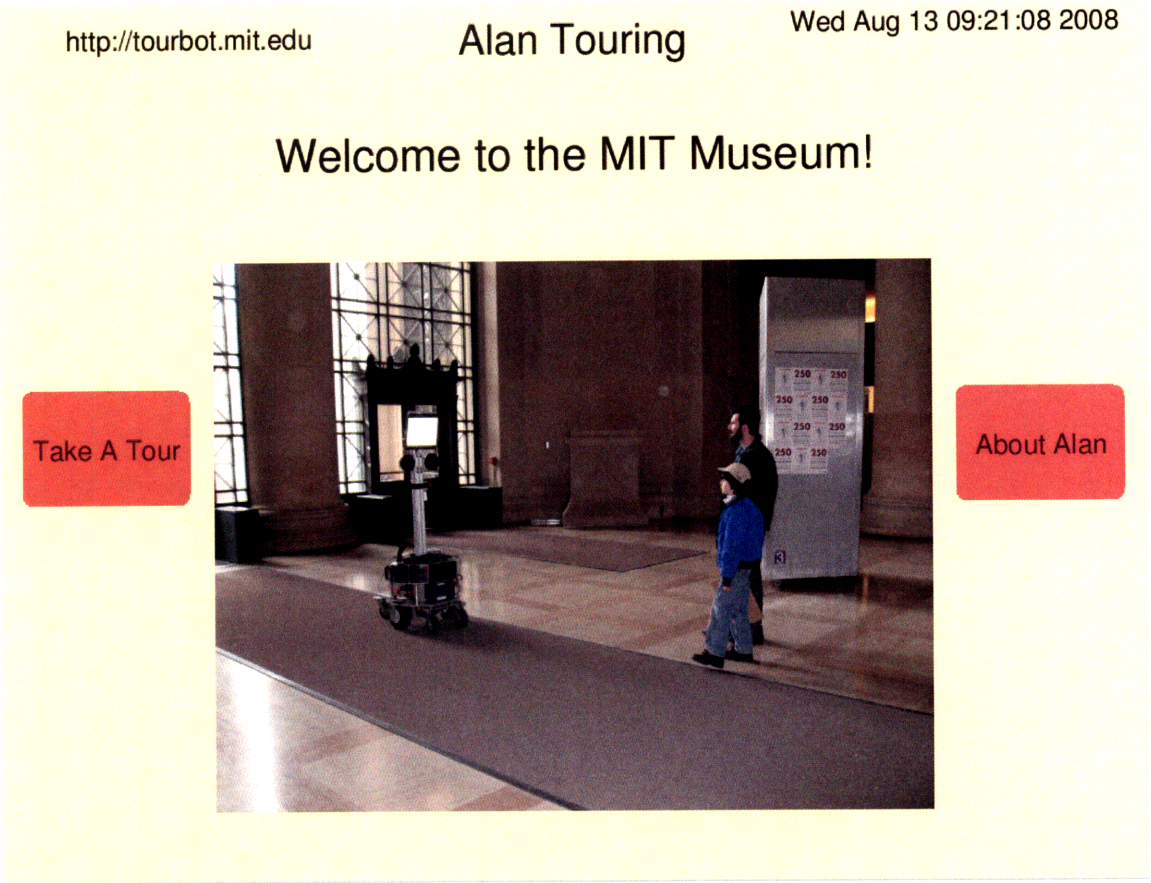


Figure 3-12: Interface currently displayed on the touchscreen.

steady most of the time.

3.6.2 LCD Interface

The robot has a touchscreen to allow for users to interact with the robot. The LCD interface is split across two screens. The touchscreen allows users to view some basic information about the robot and to start a tour when the robot is idle (see Figure 3-12). The regular LCD screen is used to display the media for the tours and a visualization of the robot's sensor data (see Figure 3-13).

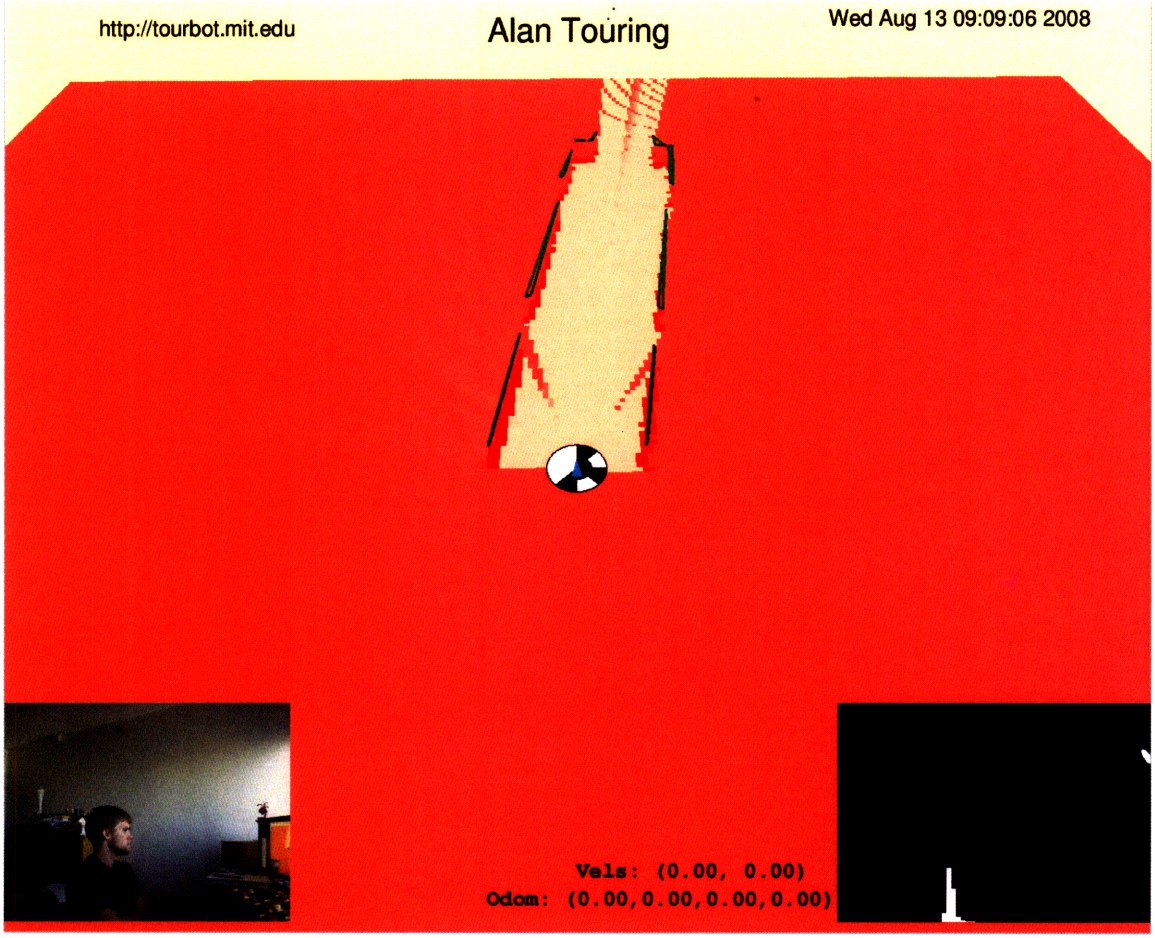


Figure 3-13: Display shown on the LCD. Bottom left corner is the face camera view. Bottom right is the ceiling camera view.

Chapter 4

System Evaluation

The overall goal of the robot was to give interactive, autonomous tours. During development, the robot described in the previous chapters underwent considerable testing on the MIT campus and at the MIT Museum. Testing in these environments subjected the robot to conditions and situations that would be expected during a completely autonomous tour. Furthermore, these tests had the robot engaged with the general public. Discussing the robot with the public provided valuable feedback on the design and capabilities of the robot. This chapter first evaluates the ability of the robot to give an autonomous tour of the MIT Museum and the describes the reactions of the public to the robot.

4.1 Tourbot At the MIT Museum

During July 2008, Tourbot spent three to five afternoons each week at the Robots and Beyond exhibits at the MIT Museum. The time at the museum was primarily spent testing the robot's functionality. Towards the end of this time, the robot was able to successfully give a few complete tours of the museum.

4.1.1 Scenario

The Robots and Beyond exhibit at the museum focuses on robotics and artificial intelligence research occurring at MIT. The exhibit is divided into five subjects: artificial intelligence, sensing, moving, socializing, and reasoning. The displays for these subjects include a vast array of robots from Marvin Minsky's robot arm built in the late '60s to Mark Raibert's hoppers to Cynthia Breazeal's Kismet. Along with the robots, each area has a television that plays interviews with researchers at MIT and has displays on the walls that describe the subjects in greater detail.

At the museum, Tourbot's goal was to lead a group of visitors through the exhibit. When the robot reached one of the displays, it would give a brief overview of the particular subject and then give a more detailed description of the robot's abilities in relation to the subject. For example, when the robot arrived at the sensing section, he would describe the lidars and cameras on the robot and then show a visualization of the map that was built by the robot.

To navigate through the museum, eleven fiducials were placed on the ceiling around the exhibit. Fiducials were placed at each display and at major junctions in the exhibit space. The largest distance between two fiducials was $8.25m$.

I accompanied the robot at all times while it was operating autonomously. If the robot needed assistance or was behaving erratically, I was able to manually override all robot navigation with a wireless joystick.

4.1.2 The Museum Environment

The environment at the MIT Museum posed some unique challenges that are not found in other parts of the MIT campus. First, the MIT Museum has a number of glass display cases that are mostly invisible to the robot – a few bolts are occasionally seen by parallel lidar, and the floor is elevated, but the angled lidar cannot detect the rise if the robot is too close to the display case. Second, the robot was barely able to travel through some narrow passages in the exhibit. Finally, the demographic of museum visitors included more children than other places on the MIT campus.

The presence of invisible obstacles posed a serious problem to the robot because the robot relies solely on its sensors for navigating through the world, as no metric map was provided for the robot. The only way the robot could safely operate with these obstacles was to keep them from being part of the ideal path taken by the robot. This task was accomplished by laying out the fiducials in the museum, so the normal path taken by the robot would keep it as far from the invisible obstacles as possible.

The hallways on the main MIT campus have a fairly uniform width that allow the robot room to navigate. The museum exhibit, though, has narrower spaces. Some of the gaps between exhibits barely have enough clearance for the robot in configuration space. As a result, the robot's path can be easily blocked, or sensor noise can force the robot to not take the optimal path between displays. Fortunately, none of these narrow passages contain objects that are invisible to the robot's sensors.

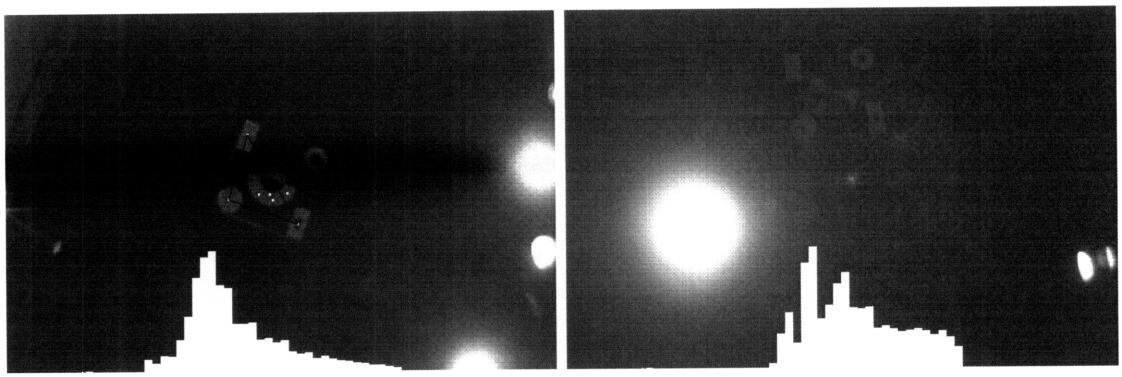
Younger children pose a challenge to the robot because they enjoy swarming the robot. A group of young children will try to climb the robot, hit every visible surface, and make sure the robot will not run them over. This swarming behavior can quickly confuse the robot and make it lose its way.

4.1.3 Evaluation

For the robot to function successfully, all of its software subsystems need to be integrated and operating correctly. During the museum testing, each of these subsystems was subjected to over 50 hours of rigorous testing. This section evalutes the performance of each subsystem and then the robot as a whole.

Communication

The communication system is essential to the robot's software. If the communication system is not working, then processes are unable to communicate and the distributed system fails. The communication system I implemented for the robot works robustly. Once the processes have established their connections, data flows between all the processes seamlessly with very little overhead. When new processes are started, they



(a) Oversaturated (b) Washed out

Figure 4-1: Frames that contain fiducials that are not recognized or are misread due to poor lighting.

are successfully linked with the rest of the system.

Localization

The localization module was responsible for determining the robot's position well enough for it to be able to successfully navigate between the fiducials, and at the museum, the robot knew its location well enough to successfully navigate through the exhibit. The combination of odometry and fiducials worked well. Initially, the robot had some difficulty moving between fiducials that were spaced further than about five meters. I discovered that the encoders had loosened in their mounts, and after gluing them in place, the odometry became accurate enough for the robot to successfully move between the fiducials without assistance.

Fiducial detection worked efficiently and reliably. The average time needed to process a 752x480 image was four milliseconds, or 250 frames per second. This fast frame rate allowed the camera to capture frames at its maximum rate of 60 frames per second. At this rate, the fiducial detection used less than 1% of the available CPU resources. The ability to run at a fast frame rate has the additional benefit of giving the robot more opportunities to find a fiducial that is in the field-of-view of the robot because more frames are processed per meter that the robot travels.

The fiducial detection was not 100% correct. Fiducials were occasionally misread when they were close to the edges of the image or the lighting was poor. The edges of the image were especially distorted because a wide angle (2.8mm) lens was used. Distortion correction was applied to alleiviate the problem, but the correction did not completely eliminate the false readings. Similarly, poor lighting could oversaturate or washout a frame, resulting in false positives and false negatives (see Figure 4.1.3). These problems were alleviated by requiring that the same fiducial be seen at least ten times before it triggered the next part of the tour.

Navigation

The navigation system needed to take the robot from one location to another along a safe route that did not collide with obstacles. An additional piece of desired functionality was for the robot to move through the world naturally, in a way that did not make people around the robot uncomfortable. In both of these goals, Tourbot was successful.

Path planning was performed using the D*-Lite algorithm, which performs a heuristic search that allows for incremental updates of the search space. Using an incremental algorithm was important to the success of the navigation system because the robot's environment was dynamic. People were constantly moving around the robot, either investigating the robot itself or other displays in the exhibit. Due to this dynamic environment, the paths generated by the navigation system were often invalidated as the robot travelled along them. When the path was invalidated, a new path needed to be found. Usually only a small portion of the map had changed, so the current path just needed to be corrected, which is why an incremental algorithm that maintains search state was a faster and better choice than a traditional search algorithm that always starts from scratch.

The paths that were generated by the path planner allowed the robot to travel as safely as possible through the world. The paths were safe because the cost map contained a gradient from the occupied region to the free region that kept paths from coming close to obstacles unless it was absolutely necessary because no other path was

available. Sometimes the robot would take a longer route that allowed it to remain further from obstacles, which occasionally confused museum guests until I explained to them how the robot decided what path to follow.

The method implemented for following the generated path successfully generates a smooth trajectory for the robot. The paths produced by the path planner were composed of line segments of varying length. When going around a corner, the path would have many very short segments to trace out the edge. If the robot tried to drive from one path vertex to the next, as was initially done, then the motion was jerky and unpredictable. By having the goal constantly moving rounded off all of these corners nicely.

The trajectory that was actually followed by the robot did not always conform very tightly to the planned path though. Usually this behavior did not pose a problem, but the robot would sometimes take a long time to move through narrow passages because it would not move tightly enough around a corner. Failing to move around a corner would then bring the robot too close to a wall, which would trigger a bump event. When the bump triggered, it would take the robot a few seconds to work itself free so that it could continue along its path.

Human Interface

Reactions to the human interface were mixed. On one hand, people reacted favorably to the eyes moving to look at the nearest face and were generally interested in the visualizations of the robot's data. On the other hand though, no one used the touchscreen interface on the robot.

The moving eyes gave the robot life. When the eyes would move and fixate on something, people would attribute that action to some internal sense of curiosity or intention, despite any actual internal states. Children would often boast to their friends, "Look! The robot is looking at ME!" when the eyes shifted to look at the child's face.

The purpose of the visualization was to allow people to see the world as the robot was interpretting it, and in this goal, it was successful. The visualization sparked

the most interest in the robot (see Figure 3-13). The visualization consisted of three parts: a three-dimensional rendering of the robot's sensor data, the image being seen by the eye cameras, and the image being seen by the fiducial camera. The rendering of the sensor data allowed people to easily get a basic idea of how the robot worked. They could see the map that the robot built, along with the obstacles it had found, where it had just travelled, and where it wanted to go. Upon learning about the visualization, people would often jump in front of the robot and ask their friends or family if "the robot could see them."

The touchscreen interface did not achieve its goal. One of the primary reasons was my presence near the robot at all times. When I was standing near the robot, people would ask me about its functionality. Rather than simply tell them to investigate the robot themselves, I would then run a process to have the robot begin giving a tour. Part of the reason for showing the robot in this fashion was the overall fragility of the system. Everything worked, but conditions needed to be correct when starting, otherwise the robot would run into trouble. The other reason was that the touchscreen interface was not obvious. It was unclear from the design of the robot that the touchscreen was actually a touchscreen. No cues were included in the interface to provide this hint without prior explanation. Additionally, the interface included little functionality so there was not a great deal of incentive for exploring the functions that the interface enabled.

Overall Performance

When the museum was not full of people, the robot was able to repeatedly navigate along a tour route that included all of the fiducials. The robot would start a tour, move to each of the important displays, and then return to the starting location very reliably. The ability to navigate through the static world shows that the robot's fundamental localization and navigation approaches work well. Due to the layout of the fiducials, the ideal path through the static world did not contain any invisible objects, which allowed the robot to travel completely unassisted.

When people were in the museum wandering through the exhibit, the robot's

performance was not as strong. The first problem that arose in the presence of people was their desire to explore the full capabilities of the robot, as described in Section 4.2.5. People would intentionally try to get the robot stuck in a corner or would try to confuse the robot by surrounding it completely. Because people were constantly trying to confuse the robot, it could no longer drive completely unaided. As mentioned, the invisible obstacles were not a problem in the static world, but when people began trying to confuse the robot, they would oftentimes force the robot's path to deviate such that the invisible obstacles now became a problem.

As a result of invisible objects, the robot never completed an unassisted tour through the museum. When giving tours, I would follow behind the tour group with a wireless joystick. If the robot became stuck or was moving too close to an invisible obstacle, then I would manually override the robot's motion to move it along a safe trajectory.

I spent much of the month at the museum trying to get all of the subsystems properly integrated, so complete tour functionality did not come online until the last few days in the museum. Furthermore, the system was still quite fragile, so a tour had to be carefully setup in order to be successful. With this in mind, the robot did lead a few complete tours through the museum with groups of about ten people following it. People generally enjoyed the experience, though it was in large part augmented by my presence. The tour had long moments where the robot was silent as it moved to a new section of the museum, and I filled in these quiet moments by talking with the group of people and by answering any questions they had.

The goal of Tourbot was to be able to give completely autonomous tours through dynamic, human-filled environments. In the end, the robot was capable of giving tours in a static world successfully, but had difficulty when the world became dynamic and full of people. Part of the problem was the way that people interacted with the robot, as they tried to intentionally cause the robot to fail in its tasks. Furthermore, one of the assumptions about the robot was that it would have the ability to sense all obstacles in its world and avoid them accordingly, but the museum environment contained a number of displays that the robot was unable to see, which meant that

the robot could not operate without supervision.

These problems show that the algorithms controlling the robot needed to be more refined to handle the dynamic and unpredicatable environment. If the robot was able to reason more about the dynamic obstacles in its world and identify them successfully, then it could have reasoned more carefully about how to proceed through the world, possibly by ignoring the dynamic obstacles and asking them to move aside if they blocked the robot's path for too long. Similarly, invisible obstacles could have been encoded into the robot's map in some way in order to keep the robot from venturing into dangerous territory.

Overall, the robot was mostly successful in its goal of being able to give completely autonomous tours. For the most part the robot did operate autonomously. I only had to step in and take control of the robot when the robot's assumption about the environment – that it could see all obstacles – failed. Given the complexity of the system and the amount of work needed to build the robot, I was pleased with the robot's abilities.

4.2 Reactions of the Public

Robots instill a sense of wonder and curiosity in people, especially children. As a robot researcher, seeing a group of children excitedly following the robot as it leads them on a tour is a true joy.

During testing of the robot over the past two years, I have had discussions with people of all ages about the robot. The reactions I observed and the discussions that I had are summarized below.

4.2.1 Driving Down the Street

The lab where the robot lives is not a part of the main MIT campus. It is located about a half mile from the main entrance to MIT at 77 Massachusetts Avenue. The only way that the robot could be tested would be to drive it down Massachusetts Ave – the main traffic artery in Cambridge – to the main campus. While occasionally a

hassle, especially during wet weather when the robot could not be tested on-campus, the need to drive the robot along a busy stretch of road proved to be enjoyable and insightful.

When people saw me driving the robot down the sidewalk, many gawked, some smiled, and others took pictures with camera phones. Tourbot did not go unnoticed, and it is plausible that, for many people, he was the first robot they had ever seen in real life. As I drove the robot, I would watch traffic on the road. Cyclists tended to slow to get a better look at the robot, sometimes they would even pull onto the sidewalk to get a closer view before returning to the street. Passengers in cars and on busses would look at the robot as they passed by. At stoplights, an occasional passenger would roll down the window and ask what the robot was.

It is hard to draw any conclusions from these observations because I feel that any strange object rolling down the sidewalk would have drawn the same reaction from passers-by. That said, the robot classifies as a strange object to most people, so they are curious about what it is and how it works.

While driving the robot, I would often encounter people working, either doing construction, directing traffic, or performing music. In almost every case, the worker would ask if the robot could do their job for them. If I could build a robot that could change city lights, pour concrete, direct traffic, play the guitar and sing, and drive a bus, then I would have made a number of people very happy in addition to having an incredible robot.

This anecdote does raise an interesting question that has been raised many times before and will be raised many times hence: what happens when robots do start replacing humans for many of these service jobs? The social, political, economic, and philosophical implications are astounding, far-reaching, and well beyond the scope of this thesis. I leave it to the reader to investigate this topic further. But one note to make is that, even if only tongue-in-cheek, the idea of robots performing general service-related jobs exists in the general public.

4.2.2 General Reactions

The observations for the rest of this chapter come from my time testing the robot in the museum and on the MIT campus.

There were two types of reactions when children saw the robot. The most common reaction was curiosity. Children were fascinated by the robot, were very curious to learn about how much the robot knew, and had strong opinions on the overall appearance of the robot.

The other reaction tended to be violent. The children would begin hitting every part of the robot and were especially enamored by the touchscreen and emergency stop button. These children would do two things. First, they would demonstrate to me the magic of the touchscreen, smiling broadly as the mouse cursor followed their fingers around the screen. Second, the children and I would engage in a little game of cat-and-mouse. The dialogue was usually along these lines:

```
Them: Can the robot move?
Me (driving the robot with the joystick): Yes
Them: What does the red button do?
Me (unwisely): It makes the robot stop.
Them (after immediately hitting the e-stop button): Make the robot move.
Me: I can't because you just hit the red button.
Them: Make the robot move again.
Me (driving the robot with the joystick): There you go.
Them (after immediately hitting the e-stop button): Make the robot move.
Me: *sigh*
Repeat the above until a parent or chaperone arrives to ferry them away.
```

Teens typically had one of two reactions: an intense interest or a cursory curiosity. Those teens that were only mildly curious might ask what the robot did, if the robot was actually a robot, or even what programming language was used for writing the software, and then wander away. The interested teens, however, would stay with the robot and asked all manner of questions. They tended to focus more on what I did

to build the robot than the robot itself. They wanted to know about the process, not the product. One teen with no technical background in computers spent over an hour discussing many different aspects of the robot, from its physical construction to how you program a robot. He did not have much exposure to technology before and the robot was a beacon to him. During the time he spent with the robot, he showed and described the robot to a number of his friends, going into detail on how long it took to build, the expense, and the robot's basic functionality and purpose. Though many of his friends displayed little interest, he was not dissuaged. He was inspired.

Adults had the most varied reactions. People with a more technical background expressed great interest in the robot, often talking with me at length about the technical details of the design and implementation. Parents accompanying children encouraged their children to ask questions. Other museum visitors expressed less interest though, glancing in the direction of the robot before wandering through the rest of the exhibit.

4.2.3 Appearance

Tourbot vaguely resembles a human. The robot's stature is that of a typical adult male; it has two eyes and ears. Each of these features serves a functional purpose. The height of the robot was necessary to fit all the hardware and allowed the LCD screens to be mounted at heights that make them easy to see. The "ears" of the robot are the speakers used for outputting audio, and the "eyes" are cameras attached to pan/tilt mechanisms so that the robot can look around. The rest of the robot is very utilitarian and consists of bare aluminum with visible wiring.

Adults and teens generally asked about the functionality and purpose of the different pieces of hardware. They were more curious about why certain things appeared where they did rather than why they were used at all. There wasn't much concern expressed about a lack of human features on the robot.

Children, on the other hand, were very concerned with appearance of the robot. As detailed in Section 4.2.3, many children asked which way the robot was facing. I tried to explain that the robot was like a tour-guide walking backwards and talking

to a group of people. This explanation did not sit well with some children at first. Once the eyes were added to the robot, it was much clearer where the robot's face was located.

The anthropomorphic features of the robot were also a problem with children because the robot did not have a complete set of human features. In particular, the robot had no arms and no identifiable mouth. The presence of these features would have fit much better with the overall aesthetic of the robot. They would not need to be actuated; they just need to be logically placed to fit the mental model that people have for a robot that is roughly human in stature.

Mental Models

My mental model when building the robot was a tour-guide that is walking backward talking to tourists, similar to the student tour guides that lead tour groups around the MIT campus. To facilitate this model, the interaction devices for the robot were placed on the "back" of the robot, where the "back" of the robot is the side that is opposite to the normal direction of travel. To me, the model is intuitive and logical in the context of a tour-guide robot. Many people, however, assume that the "front" of the robot is the direction that it is facing. The issue is further compounded because the angled lidar has a mouth-like appearance, and the back of the speakers give the appearance of eyes, even though they are meant to look like ears on the other side of the robot. In an effort to help confirm the direction the robot is facing, two cameras were mounted on pan-and-tilt platforms to act as eyes. These camera are running the OpenCV face finding algorithm, and the cameras orient themselves towards the closest face, giving the impression of attention. Figure 4-2 shows the two views of the robot.

4.2.4 Communication

Nearly every person that interacted with the robot began by saying "Hello" to the robot or waving at the robot. Unfortunately though, Tourbot does not excel at

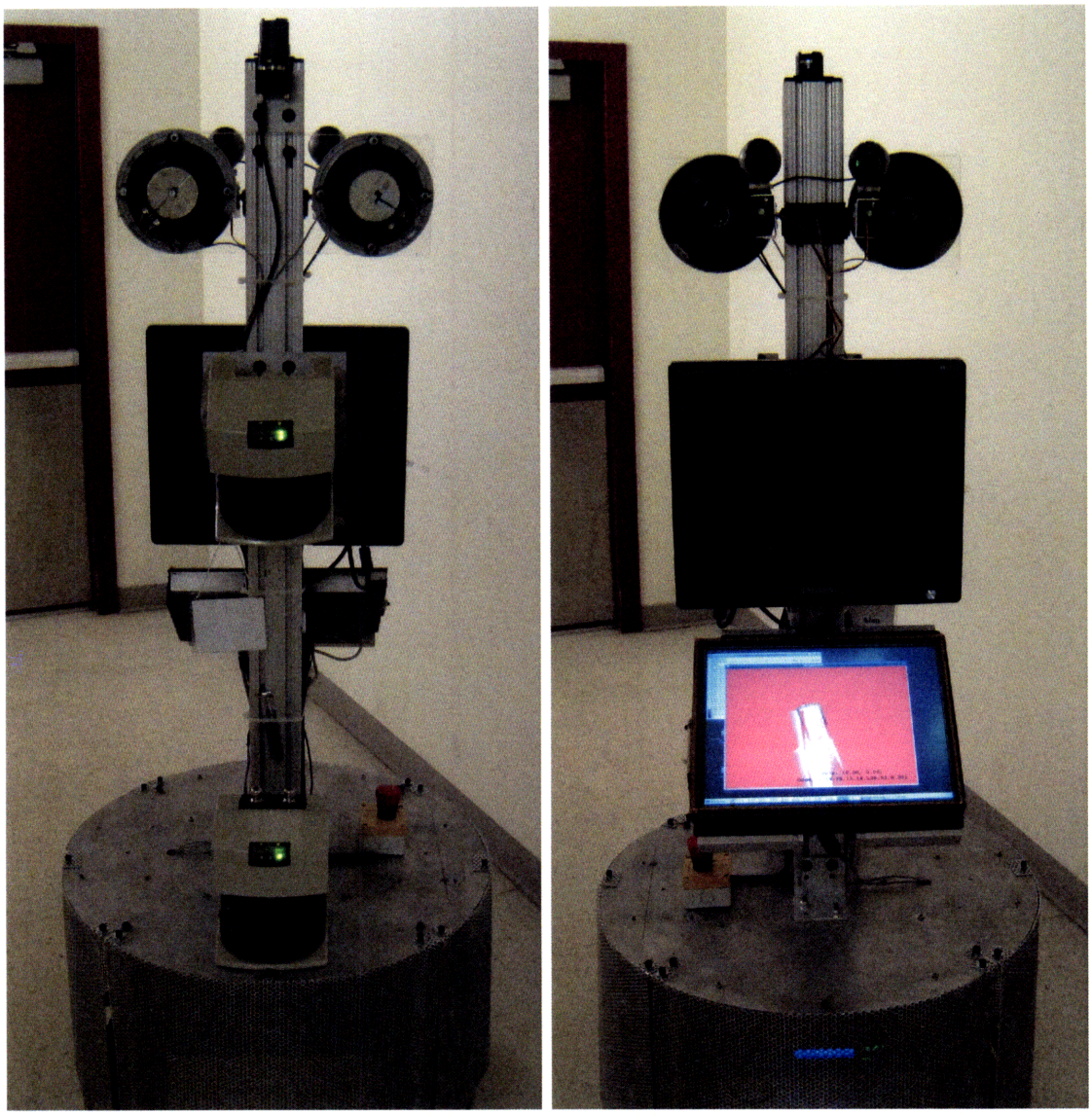


Figure 4-2: The front and back of the robot. I leave it to the reader to decide which is which.

communication. Communication with the robot is currently a one-way affair. It knows what it wants to say, and it says just that. I would explain to people that tried to talk to the robot that computers have a difficult time understanding human speech unless the environment is well-controlled, a discovery that left many people disappointed.

The desire for verbal communication was not limited to the robot understanding speech. The most asked question about the robot, aside from "What is it?", was "Can it talk?" Oftentimes during testing, the text-to-speech module was not running, so I would answer that "It can, but the voice isn't turned on right now." For teenagers and adults, this answer was acceptable, but children would continue to ask, at which point I would crumble, start the text-to-speech module, and type a few phrases for the robot to say. As soon as the robot said something, even if it was clear that it was just parrotting my typed commands, children would get excited and begin asking me to have the robot say this or that. Children would become enthralled by the talking robot. They would respond to and enforce the robot's requests, an example of which is detailed below.

At the beginning of each tour, the robot would briefly explain himself and the tour he was about to give. After this introduction, he would make a couple requests. First, he would ask everyone to walk behind him because he has difficultly moving when completely surrounded by people. Second, he told guests that he had difficulty seeing the stools that are spread throughout the museum (they are below the plane scanned by the parallel lidar and usually exist right where the robot intends to stop) and requested that guests move the stools if he is about to hit one. Once these requests were complete, the robot would begin his tour.

In one instance, two girls, maybe 9 or 10 years old, became defenders of the robot after hearing the requests. Once the tour began, they began policing the area, telling people to move out of the robot's way if someone stepped in the robot's path and moving all stools well out of the path of the robot. They then proceeded to share everything that I had told them with every other person that came near the robot. Without asking, people would learn the robot's name, that I built it, what the colors

on the sensor visualization meant, and that they should stay out of the robot's way so that he could do his job. All told, they spent over 45 minutes with the robot and had to be forcibly dragged away despite having followed the robot on a tour multiple times. These girls also debated the appearance of the robot, one thinking he was cute and one thinking he was ugly. No doubt the girls were an exception, but why?

In general, people would follow the robot's requests and try to stay out of his way and to keep stools out of his way. I was always around the robot, so they may have taken the robot's request to really be from me, or people may be willing to obey robots. Unfortunately, I can only speculate the cause. In the case of the two little girls, their debate over the robot's attractiveness is a hint that their reaction might be caused by a stronger than normal anthropomorphic mental model of the robot.

A very interesting experiment would be to investigate how willing people are to follow commands from a robot. The commands could be delivered through different modalities to see if they have an affect on willingness to perform a task. Furthermore, a comparison could be made between a human and robot by having a remotely logged-in person try to get people to follow the same commands given by the robot. Some sort of questionnaire would need to be used to determine each person's mental model of the robot which would then be correlated with the modality used for the commands and whether or not they performed the task.

4.2.5 Motion

People wanted to see the robot move. When idle, they would either pass by the robot with only a glance or stop for a quick look before continuing on their way. If I happened to be standing by the robot, those people that stopped to look at the robot would usually ask what the robot was and then continue on. If moving though, a crowd would quickly form behind the robot and follow along as it moved through the world.

A moving robot has life. It is no longer some static piece of the world; it has become a reasoning, thinking machine. Now that the machine is thinking, what it is thinking and how it is thinking become prime questions.

When the robot was moving, people were curious if it was actually reasoning about the world or following some static path. The discovery that it was actively mapping and replanning its course delighted people, and they would jump in front of the robot to ensure it wouldn't hit them, then smile happily as the robot diverted around them in order to continue on its way.

I stated in Section 3.5.3 that people stay away from the robot when it is moving jerkily and move closer to the robot when it follows smooth trajectories. My time spent at the museum justified this feeling. For the first two weeks I spent at the museum, there were bugs in the path following that made the robot come to a sudden stop, spin to correct its orientation, then start moving again. Whenever this series of events occurred, people would jump back from the robot and then maintain a further distance from the robot when the robot began moving again. I liken this behavior to driving. When a car on the road is moving unpredicably by driving between lanes or changing speeds rapidly, other drivers will give the car more space because it has violated their expectation of how cars should behave. People tend to move smoothly with gradual accelerations and decelerations, so a robot that is not moving in this fashion has violated the expectation of how things generally move. Once I fixed the bug that caused the jerking and the robot was moving along smooth trajectories, people would follow the robot from a much closer distance. The robot is behaving as people expect, so they feel more comfortable around it.

Here again, I am interpreting my observation, but this topic is ripe for an experiment. In this experiment, a robot would be navigating autonomously and a person would be asked to perform two tasks. The first task would be to follow the robot through a semi-cluttered room, which means that the robot will need to follow a twisting path to get through the room. The second task would be for the person to start at the opposite end of the room from the robot and walk along a pre-determined path. The robot would be moving in the opposite direction and would pass them at some point. Different strategies for following a path would be used, one that generates a smooth motion, another that has the robot turn in-place many times, another that has the robot accelerate and decelerate quickly. The person would perform the

task with each of these different strategies being used by the robot. At the end of the experiment, the subject would fill out a questionnaire and be interviewed about their comfort level in each of the scenarios. The goal of the experiment would be to reveal expectations that people have for a robot and how to make them comfortable in the presence of a robot.

Chapter 5

Related Work

The concept of a tour-guide robot is not new. A number of tour-guide robots have been deployed around the world in the past decade. The deployed robots have varied in their fundamental representations of the world, their approach to interaction, and their construction and design.

5.1 RHINO and Minerva

RHINO [2] and its successor, Minerva [24, 26], were two museum tour-guide robots developed in the late 1990s by U.S. and German researchers. The robots focused on robust operation in dynamic, unmodified environments. In order to achieve this goal, probabilistic algorithms and a distributed control architecture were used. At the time, computation was limited, so the robots relied on three on-board computers to handle low-level control tasks, while two off-board computers handled more intensive computation. Even then, resources were limited, so any-time algorithms, that made results available whenever needed, were employed. In contrast, Tourbot has a single on-board computer many times more powerful than the combined power of RHINO and Minerva's computers, making computation resources less of a concern.

RHINO (Figure 5-1) was the first museum tour-guide robot and was deployed in the Deutsches Museum Bonn for six days in mid-1997. RHINO used a grid-based Markov localization algorithm to find its pose in a hand-constructed map of the

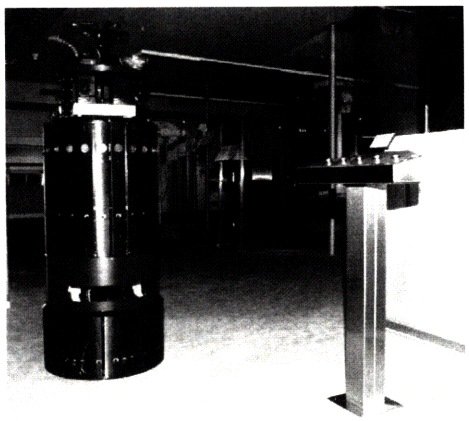


Figure 5-1: RHINO [2]

museum exhibit. The grid had a discretization of 0.15m and 0.035 radians. Range sensors were used for estimating the robot position from the map, which created a problem in the presence of dynamic objects. To solve this problem, an entropy gain filter was applied to the sensor readings. The basic functionality of this filter is to not use a sensor measurement that decreases the certainty of the robot's position.

Figure 5-2: Minerva [26]

As the follow-up to RHINO, Minerva (Figure 5-2) was deployed at the Smithsonian Museum of American History in Washington DC in the fall of 1998. While Minerva and RHINO shared much of the same overall software framework, Minerva introduced advances in automated mapping and localization. Additionally, Minerva focused more on human-robot interaction than RHINO.

RHINO[2] required a hand-built metric map that took nearly a week to build. One goal of Minerva [26] was to solve this problem by applying expectation-maximization (EM) mapping, an off-line simultaneous localization and mapping (SLAM) variant [28]. Using this method, a map could be built in a matter of hours. Tourbot does not use a pre-built metric map, but instead relies on a topological map of fiducials for finding its way. The process requires placing and measuring the distances by hand, akin to the hand-built map used by RHINO.

The museum where Minerva was deployed was much more open than RHINO's museum. Two strategies enable Minerva to remain localized in this more challenging environment. First, a mosaic of the ceiling was created. A camera aimed at the ceiling augmented the laser-based localization. Second, a motion planning strategy known as coastal navigation [22] was used. In coastal navigation, the entropy of a pose is added as a state variable. By augmenting the state, the planned paths keep the robot close to recognizable features to help it stay localized. Additionally, Monte Carlo localization (MCL) using particle filters [27] was used for determining the actual robot pose. MCL estimates the robot in continuous space and is much more efficient than the grid-based localization used by RHINO. These approaches are not needed for Tourbot because the decision was made to avoid trying to localize to a metric map.

Minerva also placed a greater emphasis on human-robot interaction [24]. A 4-DOF face comprised of eyes, eyebrows, and a mouth was used to convey the robot's intent and to try and draw new guests for tours. The face was tied to an internal emotional state that would change when the robot became blocked. Tourbot lacks the fully animated face, though its eyes are used to convey interest in its users. A similar internal state based on whether Tourbot is blocked or not is also maintained.

5.2 CMU Mobots

The museums at Carnegie Mellon University (CMU) have been home to a number of tour-guide robots deployed for long-term operation [31]. The focus of the Mobots study was on building reliable robots that would operate autonomously with no supervision. To achieve reliability, the robots operated in engineered environments (fiducials were placed on the walls), and the robots were constrained to stay within a certain distance of a fixed path. They were only allowed to deviate a small amount to avoid obstacles. If the avoidance behavior would take them beyond the acceptable range, then they would ask the obstacle to move.

The Mobots were extremely reliable robots and experienced few failures. The

(a) Chips (b) Sweetlips (c) Joe Historybot

Figure 5-3: The CMU Mobots [31]

reliability was due to the design philosophy of the researchers whose mantra was "simplicity in design and paranoia in implementation breeds confidence in deployment." To help avoid failures, a simple principle was applied to all of the robots' actions. Every time a task is performed or a command is issued to hardware, check that the operation was successful. If it failed, then try the task again or reset the hardware. The "try-again" principle was used because a robot's environment has enough nondeterminism that running the same operation twice can have different outcomes. Aside from the extremely reliability, the Mobots experiment placed robots in the environment for years, resulting in a number of lessons learned about the nature of human-robot interaction.

The first lesson was that public robots need to be able to handle physical abuse. They found people like to try to damage robots and make them malfunction. My experiences with Tourbot corroborate this fact. Even when supervised, people will try to force the robot to fail, though physical abuse didn't happen because I wouldn't let it. The researchers who build RHINO and Minerva also reported that people would try to force the robots into uncharted territory including areas with robot-killing stairs [2, 26].

The second lesson focuses on keeping users' attention. The researchers found that

people will not pay attention to robots for a long period of time if no interaction is happening. The first Mobot, Chips, played two-minute videos at various stop in the Dinosaur Hall at the Carnegie Museum of Natural History and would quickly lose the attention of guests. The later Mobots all had increasingly sophisticated dialogue systems to try and maintain interest during interactions. A specific recommendation that they make is to ensure that the robot has enough dialogue to avoid repeating itself quickly. As soon as a repeat happens, users become uninterested in the robot. With Tourbot, I was always nearby and during periods where the robot was moving from one display to the next, I would talk with the guests following the robot to keep them engaged.

Another finding related to interactions was that people generally treat a robot as part human, part machine. Initially, people treat the robot as a human and then adapt their behavior depending on the person's goals. For example, people will generally get out of the way of a robot at first, but on further investigation, may start intentionally blocking the robot to discover how it reacts. This finding is inline with museum visitor's interactions with Tourbot. As mentioned, almost all guests would try to initiate a verbal dialogue. When that approach failed, they would view the robot more as a machine or curiosity than a person.

5.3 RoboX

The previously mentioned robots were all developed in a research context. RoboX [30], on the other hand, was designed from an industrial standpoint. The goals established for RoboX similar to other tour-guide robots: safety for human and the robot's environment at all times, fully autonomous and reliable navigation in unmodified environments, and multi-modal interactions.

The robot was built by the Autonomous Systems Lab at EPFL and BlueBotics SA, a robotics company aimed at putting robots in public. The physical appearance was designed by artists, industrial designers, and scenographers, resulting in a robot with a clean, unified exterior (Figure 5-4). In stark contrast is Tourbot, which is a

mess of wires and bare metal.

Figure 5-4: RoboX [30]

The industrial concern with safety shows in the control system for the robot. All critical tasks – obstacle avoidance, motion control, planning, and localization – ran on a real-time operating system XO/2 as opposed to the more typical Linux operating system in research robots. Furthermore, a separate microcontroller controls the operation of the motor with a hard-programmed speed limit. Commands from the motion controller are sent to the microcontroller. If a command is not received less than 0.2s after the previous command, the robot is immediately halted because condition indicates a crash on the navigation computer. Tourbot uses the Linux operating system with no real-time patches. The motor controller board used by Tourbot includes a watchdog functionality that halts the robot if a motor command is not received within one second of the prior command.

From an industrial standpoint, the robot went through several well-defined stages of development. All mechanical, esthetic, and electrical design was finalized within three months of the project start. Two test robots were built and tested, then a small redesign phase occurred, and the final robots were constructed. Extreme programming philosophies were used during software development. A strong emphasis was placed on object-oriented design to allow for fast replanning cycles and integration of new code. Extensive testing of the navigation system was performed and initial results showed a MTBF of 4.4 hours. After continued work, the MTBF had increased to 20.4 hours during the first month of operation.

In comparison, no esthetic design was performed for Tourbot, and the mechanical and electrical systems have been in a constant state of flux. The very basic shape of the chassis has remained the same, but the location of hardware has changed several

times, and the robot has been rewired multiple times as well. The software design followed object-oriented principles, but the implementation remains fragile. Extreme amounts of programming were done, but specific implementation philosophies were applied beyond design it, write it, test it, rewrite it, retest it, etc.

While I have never performed any actual testing of the MTBF because the robot development has been constant, I'd estimate the MTBF to current be around 15 minutes. By that time, the robot will have come dangerously close to an invisible object or been deviated from its path such that it gets stuck in a corner.

5.4 Robust Robots

Previous tour-guide robots have primarily existed in museum settings. These robots focused on reliable, robust, and safe navigation through their environments to protect the robot, the exhibits, and the people. Each robot solved these tasks in different ways. The interactions with people were simple and involved a user responding to basic questions using either a touchscreen or a set of buttons mounted on the robot. In each case, the robots were able to operate with complete autonomy for long stretches of time.

The goal of Tourbot was an ambitious expansion on these previous tour-guide robots. Tourbot was going to operate in a less controlled and much larger environment than previous robots. Furthermore, Tourbot was going to be capable of much more complex interactions. More complex interactions were needed because the human tour-guides' main purpose was to answer questions about the MIT campus, which meant that the robot needed this fundamental ability at well.

Tourbot did not achieve the initial goals (though development is still underway), but did make considerable headway in reaching the goals. The robot had a somewhat successful deployment in a museum setting, like the previous robots. This deployment revealed the importance of making the correct assumptions about the world and being able to robustly handle unforeseen situations. For example, the museum setting contained corners in which the robot could easily become stuck, and it contained

invisible obstacles that the robot could hit. Tourbot did not contain the sensing capabilities to match its assumptions about the world.

Previous tour-guide robots solved these problems in a number of ways. RHINO and Minerva existed in environments where not all obstacles could be seen. They augmented the world with a map fake sensor readings to get around this problem and employed fundamentally probabilistic approaches to all aspects of the system. The Mobots used a very different approach and used very simple world models that were coupled with sensors capable of detecting any obstacles that might arise in these basic worlds. In each case though, the sensing abilities of the robot matched the model used for the world. All of the robots were able to detect when they were unable to progress due to a dynamic object versus some static feature of the world.

My work and work on previous tour-guide robots shows that operation of robots in dynamic human environments requires the builder to make assumptions about the world. The robot needs the ability to sense all obstacles with the constaints of the assumed world. This sensing will allow the robot to correctly operate within these constaints. For complete autonomy though, the robot also needs to be able to detect when its model of the world has failed. For example, if Tourbot had been able to recognize and differentiate a dynamic obstacle from other obstacles, then it could have successfully navigated the museum on its own. When the Mobots deviated from their defined paths by more than some preset tolerance, they would recognize the situation and stop until they could return safely to path they were designed to follow.

Chapter 6

Learning From Tourbot

At the time of this writing, I have spent the past 31 months designing, developing, and testing the robot described in this thesis. After the initial phase of development during which the basic hardware platform was constructed, I have spent the majority of my time working alone on the robot software. When problems with the hardware were discovered, I would work with the hardware team in developing a solution. During the robot's lifetime, there have been clear successes and failures in both the design approach and implementation of the robot. This chapter considers some of the choices that were made in designing and implementing the robot, focusing first on experiences with the robot hardware, followed by a discussion of the software, and finally an explanation of lessons that I learned while working on the project.

6.1 Hardware

6.1.1 Chassis

The majority of the work on the chassis was done in the first four months of the project. During this time, the chassis was constructed with an emphasis on expediency over careful design. In the rush to get a driving robot, analysis on the weight distribution of the robot, consideration of the exact parameters of the robot's environment, and calculation of the space needed for the sensors and hardware were

skipped. Each of these oversights caused headaches and inconveniences during later stages of the development process.

Unsteady Wheels

During our initial design meetings, the Tourbot team selected to use motors with drive shafts that are perpendicular to the motor body because they would allow the bulk of the robot's weight to sit below the motor axles. However, these motors were out-of-stock when the time came to purchase them. Rather than wait until the motors became available, inline motors were purchased because they were available at the time. This decision caused a massive overhaul in the design of the robot. Now the motors were to be mounted beneath the bottom plate, shifting height of the bottom plate higher by 0.15m. At the time, the concern was more cosmetic because the team was trying to avoid having another "rolling garbage can" robot, with apologies to R2D2. Once the chassis was constructed though, it became clear that the robot was not entirely stable. Wobbles were noticed when the robot stopped or travelled over small bumps. One day while driving the robot back to our lab, a front caster wheel dropped into a divot and the robot flipped over. Fortunately, no sensors or delicate hardware were mounted at the time and no one was hurt. At this point, the best thing to do would have been to go back to the drawing board while the chassis was clean and the mounts for the sensors had yet to be built. Instead, more care was taken when the robot was being driven, but no hardware changes were made. Eventually, the instability could not be avoided and the problem had to be addressed when the number of potential solutions was much narrower. Section 2.1 discusses the solution that was ultimately implemented.

Shifting the bottom plate up 0.15m placed the batteries, by far the heaviest objects on the robot with a combined weight of 68kg, above the wheel axles. The tipping problem then occurred because all the weight of the robot sat above the wheel axles and the caster wheels did not provide enough support. When the robot stopped suddenly, a rotational moment of inertia was above the axles creating a torque capable of tipping the robot. Figure 6-1 shows how the robot would tip.

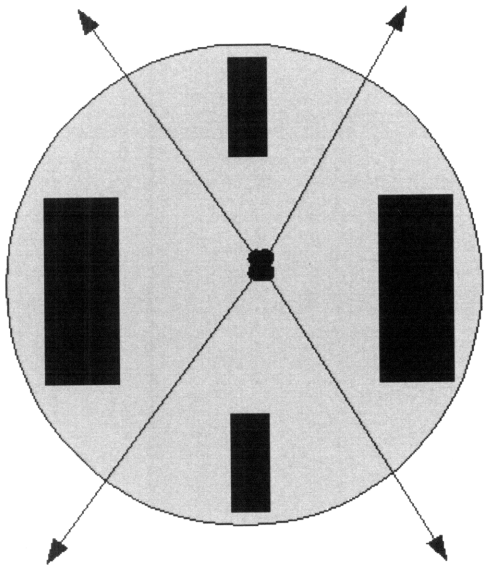


Figure 6-1: Vectors along which the robot would tip

Breaking the Circle

Section 2.1 describes the final chassis of the robot. The biggest difference between the final chassis and all design goals mentioned so far is that the final chassis was not a circle. The robot had a more hexagonal shape and looked a bit like a lollipop with a short handle (see Figure 2-1). This solution was implemented because it was the easiest option available, but in retrospect, breaking the circle created many more difficulties.

From day one, the plan for the robot was to have the metal chassis with a touch-sensitive bump skirt mounted around it to ensure that the robot would not run people over or cause damage to its environment. The bump skirt would activate from all directions using a mounting system similar to James McLurkin's Swarm [18] and was to be built from fiberglass. Using fiberglass would allow a thin skirt to still be strong and damage-resistant. Once the circle was broken though, creating a bump skirt became a much greater mechanical problem. A number of ideas were brainstormed for constructing a bump skirt and one prototype was built, but no viable solution was ever found, and the robot was forced to rely solely on its lidars for not colliding with objects in the environment. The reliance on lidars, as described in Section 6.1.2,

proved to be unsatisfactory.

Algorithmically, a non-circular robot causes two major difficulties. The most obvious difficulty with a non-circular robot is that the lack of radial symmetry adds a dimension to the configuration space, shifting the space from $\Re^2$ to $\Re^2 \times S^1$. Adding the extra dimension makes the robot navigation problem more difficult. The planner now has to consider the orientation of the robot in order to be complete, or in the case of a discretized representation, resolution complete. The difficulty is further exacerbated because the generated path now includes specific (x, y, θ) waypoints, which makes control laws for getting the robot to follow the path more difficult.

The alternative to expanding the dimensionality of the planning problem is to approximate the shape of the robot as a circle by using the radius of the circle inscribed when the robot spins around its rotational axis. This strategy provides the simplicity of planning and controlling a circular robot at the cost of completeness. The planner now considers accessible places to be inaccessible. For smaller robots, this tradeoff might be acceptable because the slight increase will not severely inhibit the robot. However, Tourbot was a large robot, with a diameter of 0.66m, and with the wheel extensions, the diameter of the circle approximation for the robot was $1.2m$. The robot now needs almost half the width of the Infinite Corridor free in order to navigate. Despite these problems, I used the circular approximation for the robot, as described in Section 3.5.2. These effects of this decision are explained in Section 6.2.3.

The other major difficulty of a non-circular robot is getting it out of trouble. The robot will not perfectly navigate the environment. It is the nature of robotics and one of the key difficulties in building a robust robot. For example, people enjoy trying to confuse robots in much the same way that a little brother enjoys pulling his sister's pigtails when she is sitting in front of him in a car. You can't explain why you do it, but it is something that everyone does. The result of this robot harassment is that a robot can end up in many positions beyond where it was intended to be, with corners or niches in walls seeming to be a favorite of Tourbot.

Tourbot determines if he is stuck or cannot proceed based on the distance of the

nearest object. If an object is too close, then the robot stops – think of it as an optical bump skirt. Once stuck, the robot has to figure out how to get unstuck. At first, he waits to see if whatever is blocking his path is only a temporary obstruction, like a person passing to close to the robot. If the obstruction is not temporary, but is instead a petulant teen or a devious wall, then the robot needs to extract itself. Two options present themselves: rotating in place or backing up.

Rotating in place is a surefire way to escape trouble for a circular robot. The robot can rotate until its sensors determine a clear path, and then it can proceed. The process is simple, safe, and if there is a path the robot can follow, then it is guaranteed to find it. Spinning in place is not always an option for a non-circular robot. When spinning, an object that did not previously impede the robot can become a road block. At the MIT Museum, there were many occasions where the robot would manage to get wedged between an exhibit and a post or guard rail and spin back and forth in a heartbreakingly fruitless attempt to free itself. That said, there were many times where a simple rotation freed the robot, thereby allowing it to continue on its journey.

If a robot cannot rotate or travel forward, then the only other direction to go is backwards. For some robots, going backwards is no different than going forwards. Unfortunately, Alan Touring is not one of those robots. The reason is that he has no sensors the look behind him, which means that in reverse, he is travelling blind. One way to backup is to simply trace the route that was followed into danger. The big problem with this approach is that the robot has people following him, so the path just travelled is now likely to be blocked. Furthermore, because backing up is a scary option, the robot always tries to rotate free first, so backing might carry the robot into a wall or display case. Ultimately, a stuck robot is an extremely unpleasant situation. Therefore when the robot becomes stuck and cannot rotate free, it will announce that it needs to backup because it seems to have gotten stuck. At this point, it moves backward very slowly until it detects that it can move freely again, then begins its journey again – a necessary, but unsatisfactory solution.

6.1.2 Sensors

Using encoders for dead-reckoning, lidar for mapping and obstacle avoidance, and a camera for reading fiducials were simple choices. Encoders are essential for any wheeled robot system. Lidar is a far better range sensor than IR, sonar, or radar. The fiducials were designed to be read by a camera. Other sensing choices, however, caused a number of headaches as will be revealed below.

See-Through Walls

People shouldn't throw rocks at a glass house, and robots shouldn't be in a glass environment. Why? Glass is invisible to lidars. The robot relies solely on lidar for all obstacle detection, making its operation around glass nerve-wracking. As mentioned in Section 4.1.2, the MIT Museum contains a number of glass exhibits. Similarly, glass walls are becoming much more popular on the main MIT campus, with the majority of the architecture department employing glass walled offices and studios for example.

The robot could survive in glass environments if it had a bump skirt or some sort of preloaded map of inaccessible areas. With a bump skirt, the robot would be able to detect when it was colliding quickly and could stop before any damage is done. Furthermore, the space occupied by the robot when the bump occurs can be added as an occupied region of the world so that a quick replan will hopefully avoid the glass area. If the robot had a map of the inaccessible areas as related to the fiducials, then these areas would be avoided as the robot planned its route through the world. A ring of sonars may also be sufficient for detecting glass, but glass is almost a perfect reflector for sound, so the number of sonars required to ensure the angle of incidence with the glass is small enough is prohibitive.

6.2 Software

Software on the robot was continually evolving as new capabilities were envisioned and implemented. The distributed development approach that was used with the robot

and allowed the robot's capabilities to expand without affecting previously developed – and functioning – pieces of the system.

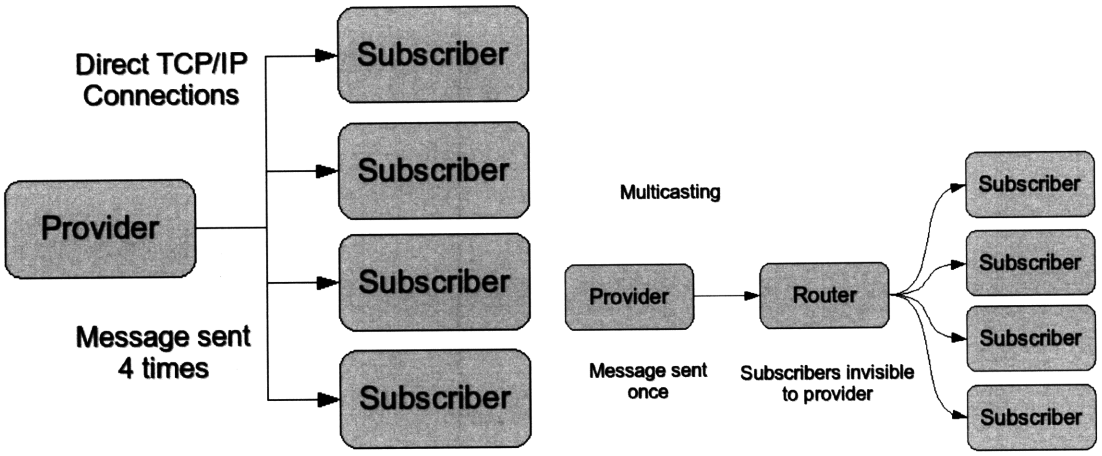
6.2.1 Communication

The communication system that I wrote for the robot was successful. It allowed for computers and processes to be dynamically added and removed from the system without failing. The interface was easy to use and the overhead of transmitting messages was very low. Once the system was working, very little additional work was needed on the part of processes to transmit and receive data. There was, however, one aspect of the system that could have been made more flexible.

Whispering Instead of Yelling

As described in Section 3.2, the communication system works by establishing direct TCP/IP connections between processes. Each process that desires a certain type of data connects to a process that provides it. Processes that produce data used widely, like odometry or lidar readings, will be sending data to many other processes many times a second. TCP/IP is not the best means of accomplishing this task because TCP/IP connections are a one-to-one connection, meaning that the data being sent has to be re-sent to each individual connection rather than a single transmission that reaches every process. It is the equivalent of telling a group of people the same thing by whispering in each person's ear, as opposed to yelling it for all to hear. There exists a way to yell across a network, though.

Multicasting allows a computer to shout its message to the world, or at least the network. Multicasting allows for write-once, read-many semantics. The idea of multicasting is that the router handles the task of sending the packets to all interested processes, rather than placing the burden on the transmitting process. The bandwidth needed by the processing sending the data is drastically reduced in the case of multiple subscribers and the overall flexibility of the system is much greater. The added flexibility occurs because multicasting uses connection-less UDP packets, so processes



(a) Direct TCP/IP

(b) Multicasting

Figure 6-2: Direct TCP/IP connections vs. UDP Multicasting

that are receiving data are completely unknown to the senders.

To implement this change, processes would now connect to DataManager to be informed of the multicast address for the various data types that they wish to receive. The one problem that exists here is that multicasting across a global network may not work, so special processes on each subnet may need to forward data between subnets, a task that is completely feasible.

6.2.2 Localization

The localization system performed well because the decision was made early to avoid any sort of global localization. The fiducial system was designed to keep the robot from needing to travel more than 10m accurately. In doing so, the robot was able to rely solely on encoder-based odometry. While this system worked well, the effort needed to produce and place the fiducials makes the system rigid and unwieldy. Recent research has developed systems that could eliminate the need for fiducials, though they are admittedly much more complex. A stop-gap approach is to reduce the number of fiducials that are needed. This approach requires more accurate odometry than encoders alone can provide. The following describes the elimination or reduction of

fiducials.

An Engineered Environment

Tourbot relies on an engineered environment for localization. Fiducials were used for localizing the robot because they are a simpler to use than methods that try to directly measure the world. Recognizing a fiducial on the ceiling is more reliable than trying to recognize a patch of ceiling, but is also less flexible. In order for the robot to be able to operate, considerable time must be spent constructing, placing, and measuring the fiducials. For the MIT Museum, a total of 11 fiducials were used and the whole process took about five hours. Scaling this process up to an area the size of the Infinite Corridor that requires well over one hundred fiducials would be daunting. However, tour-guide robots have been successful in localizing in unstructured environments, showing that a fiducial-based approach is not necessary.

Minerva [26] used a probabilistic localization method that depended on matching lidar scans and images of the ceiling with a previously built map. Since that time, the emergence of the scale-invariant feature transform (SIFT) algorithm [17] has led to exciting developments in vision-based simultaneous localization and mapping (SLAM) [25, 6]. These vision-based SLAM systems could potentially allow the robot to operate without fiducials. Instead of placing a series of fiducials in the environment and establishing a relative map, a SLAM-based approach would require driving the robot through the environment in question a limited number of times so that it could build an accurate map. Locations in the produced map could then be tagged with tour content. This localization system is much more flexible because the robot is able to operate in a new environment by being manually driven through it and then by tagging regions of the map with appropriate content.

The fiducial-based localization was chosen because the initial goal for Tourbot was to have a functioning robot within a year, and SLAM-based approaches are difficult to get functioning reliably. The robot has been under development for much longer than the original year though, and if I had known how long I would be working on the project, then I would have used a more flexible localization method that did not

rely on an engineered environment.

6.2.3 Navigation

The navigation system successfully moves the robot through the world. If a path to the given goal can be found, the navgiation will find it and guide the robot the position. Errors in arriving at the desired goal are caused by odometry error. Furthermore, the global planning through the fiducials is able to handle errors due to a missed or incorrectly read fiducial or unknown starting location. When the robot gets off-track, the robot does not fail, but re-routes its path so that the tour can continue successfully.

A Reverse Curse

The curse of dimensionality is the exponential increase in the state space of a problem for each added dimension. For robotics, each degree of freedom searched expands the state space. With Tourbot, I used a two-dimensional state space, performing searches in $\Re^2$ to reduce the computation time. To perform these searches, the robot is treated as a large circle as mentioned in Section 3.5.1. Treating the robot as a circle expands the footprint of the robot from $0.58m^2$ to $1.13m^2$. The robot is already large enough, so doubling the footprint makes the navigation problem much harder. Searching in three-dimensions, (x, y, θ), would increase the computational complexity of the search, but provide a great deal more flexibility. For example, it is currently impossible for the robot to travel autonomously through a doorway.

One approach to expanding the search space to three dimensions is to use brute force. To implement this approach, the cost map is converted to a three-dimensional grid, where the third dimension is the discretized orientation of the robot. The shape of the robot here would be a rectangle or hexagon, and an update to an occupied cell in the cost map would now involve rotating the rectangle to each orientation and changing the costs for the corresponding cells. To search this space, the successors of a search node now become a more complicated function of the current robot configu-

ration and the kinematic constraints of the robot. This approach can be complicated and potentially very computationally intense.

An alternate approach that I designed but did not have time to implement uses the actual configuration of the robot, but still searches in a two-dimensional space. To accomplish this task, the successors of a search node consist of a set of motion primitives. The primitives are arcs that the robot would move along. Because the motion of the robot along the arc is known, the parts of the world that the robot will travel through can be pre-calculated. As a result, the map can still be in two dimensions and all possible collisions will still be correctly considered. Furthermore, the search can happen in a continuous representation of the world because the arc primitives are not discretized, but instead represent the actual coordinates that the robot will occupy while travelling along the arc.

Further alternatives include various sampling-based search methods like rapidly-expanding random trees (RRTs) or probabilistic roadmaps. These techniques have been successfully used on very high-dimensional kinodynamic planning problems, so they would work well for finding a route for the robot through a dynamic region. Their ability to scale to higher dimensions allow even more flexibility for potential searches. One possibility would be to search in a state space of the robot that includes the velocity and acceleration for much more refined control.

6.2.4 Human Interactions

The weakest part of the robot is its ability to interact with people. The only interaction people can have with the robot occurs using a touchscreen. The touchscreen interface, as described in Section 3.6, allows people to start a tour, find out about the team that built the robot, and sign a guestbook. That's it. During tours, the robot just plays back scripted slideshows. A robot that aims to be social needs to be much more interactive.

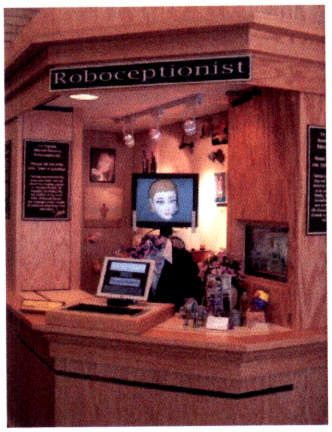


Figure 6-3: Valerie the Roboceptionist [13]

A Deaf Robot

When people first encounter the robot, they typically wave or say "Hello", but the robot has no idea they are performing these actions. Instead, it just sits there blankly. As the robot's shadow and protector while it is in public, I quickly step in and let people know that the robot can't hear them because "computers aren't very good at understanding people talking." Revealing this fact usually sparks a series of questions about what the robot can and cannot do. Generally, people expect that a robot should be able to understand their speech and react intelligently to it, which is far from the current state-of-the-art. I can only guess that this conception is propagated by the depiction of robots in Hollywood films.

Despite the lack of speech recognition, complex interactions can be performed between robots and humans. In [13], people communicate with a robot using text typed on a keyboard. By using a keyboard, the complicated speech issue is bypassed. Valerie attempts to have a dialogue using a chatbot called Aine. The researchers found that people would generally stay for one or two sentences of dialogue and then move on, especially if the robot begins to go into a long monologue. The researchers encouraged future social robots to be interactive to maintain user interest.

With Tourbot, potential interactions would be short-term with tourists as they find out information about the campus or tell the robot they'd like to take a tour. Even with the short-term nature of the interactions, some sort of dialgoue capability

would be important for giving a more natural feel to the interaction. Furthermore, the dialogue system should try to incorporate dynamic elements, like the weather or the current emotional state of the robot.

Your Friendly Neighborhood Robot

Due to the problems discussed above, Tourbot is not a very human-like robot. He does not possess any internal representation of emotion and does not respond to his environment beyond asking people to move out of his way when they encroach in his personal space for too long. The monologue delivered by the robot contains bits of wry humor only because it was written by me. I would much prefer for the robot to be capable of dynamically shifting his mood and tone of voice based on his interactions and experiences for the day. There would be a set of weighted stimuli that would determine the robot's mood.

The mood of the robot could be conveyed using dialogue trees in a manner similar to many role-playing video games (RPG). A common interaction in a non-player character (NPC) in an RPG involves a scripted dialogue, where the response of the NPC is based on a number of factors, including the responses of the player, the time-of-day, the environment, and past interactions with the player's character. A robot's interaction with people is more dynamic than in an RPG though, so a pre-selected set of responses to questions or comments by the robot would quickly be tiresome. People want to have conversations with the robot. An emotion system could allow the robot to become more interactive by providing a basis for navigating through a dialogue tree. For example, a lonely robot would respond very differently to a greeting by a person than an annoyed robot. The emotion of the person interacting could also be taken into account by using the ConceptNet system [9], which uses a common-sense reasoning engine to try and guess the emotion of a piece of text. To get more interesting sentences, basic responses would be provided to the robot. These responses would be tagged using a natural language processing system. Now, when the robot is formulating a response, it can search through a list of words tagged by the natural language processor and ConceptNet. New sentences could be then formulated

by replacing generic words with their more emotionally-charged alternatives.

6.3 Lessons

This robot is, to date, far and away the largest project that I have built. As such, much of the development time was a learning process in how to engineer a large and complex system. Many of the design principles and engineering practices taught in classes were useful, or would have been useful had they been applied. This section discusses general practices that, had they been used, would have resulted in a much smoother development process.

6.3.1 The Team

Robotics is a multi-disciplinary field that requires expertise in a number of domains including mechanical, electrical, and software engineering. Few people have strong skills in all of these arenas which makes a team crucial for successful development. In building a team for a large project like Tourbot, a major consideration should be the amount of time building a robot takes. Putting a robot in a the real-world and having it perform robustly is a very difficult and time-consuming task, so team members should be prepared to – and able to – spend long hours debugging the bevy of problems that are guaranteed to arise during system development. Furthermore, regardless of how much time is spent in the initial design, conditions will arise when the robot is tested and deployed that were never considered. As a result, the time commitment for a robot extends over the entire life of the robot rather than a big initial push and then only occasional tweaks.

The skill of the team members is important, but less so than motivation about the project and the patience for debugging. Motivation and patience are so important because of the time that will inevitably be spent debugging baffling problems. One example is a recent bug that caused the entire electrical system of the robot to fail. For unknown reasons, the robot refused to power on and the symptoms pointed to a short-circuit in the wiring. I checked every wire on the robot to see if it was

improperly grounded to the chassis and found nothing. Ultimately, I opened the DC/DC converter and adjusted a few potentiometers and the robot magically powered on again. From first symptom to solution was about eight hours. Additionally, a motivated team will learn and evolve as the project carries forward and will not be daunted when mistakes are made that cost time and money to fix. Experience is the most valuable teacher.

In general, there will be one person that is the overall leader of the team, a role that I assumed for the Tourbot. This person needs to be prepared to handle all of the headaches associated with running a project: finances, deadlines, and team management. Each of these pieces requires considerable effort.

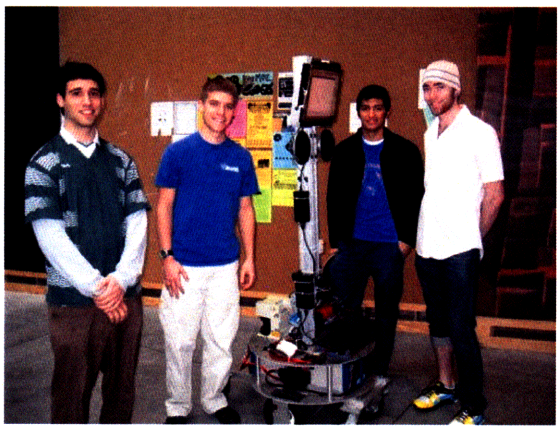


Figure 6-4: Tourbot team members (Left to Right): Behram Mistree, Collin Johnson, Patrick Barragan, Will Bosworth

The finances of the team need to be carefully watched to ensure that there is enough money to purchase the necessary parts and to pay for services, like machining. It is very easy to spend a lot of money on parts, only to realize that they do not work as you thought they would, or require a high-level of maintenance. These situations can quickly consume a considerable portion of the budget. Being careful in acquisitions will certainly payoff in the long run. Another temptation is to provide food at all team gatherings, but buying lots of a food is a surefire way to burn through money, and should be avoided except in very special circumstances, like a kickoff meeting for a new phase of the project.

Deadlines are a necessary piece of the project because they help motivate the team toward a specific goal. I discovered that realistic, short-term deadlines are far more valuable than lofty and ambitious long-term goals. Successfully reaching a deadline provides a good morale boost and spurs development, whereas a failed deadline can

be frustrating and discouraging. I personally set far too many personal deadlines that ended up being completely infeasible and led to a considerable amount of wasted or misguided effort because the deadlines forced me to implement hacks that would end up staying in the system for too long. If I had followed the original design and paced myself properly, then actual development would have been much smoother because the pieces would fit together better. It's similar to the difference between holding something together with duct tape and holding something together with bolts. The duct tape will last for a few days or weeks, but it will fail soon enough. Bolting something in place, however, will ensure that it stays holds for the long-term.

Team management is a delicate task and depends entirely on the dynamic of the team. For the Tourbot project, I found myself forced to constantly monitor the status of all pieces of the project, figure out what work needed to be done, and prod people into actually doing the work. The biggest problem that I encountered was a lack of initiative on the team. If I specifically asked people to solve a problem, saying making a mount for the touchscreen, then they would do it. But once the work was finished, that would be the end of it. Questions like, "What can I work on now?" or "I thought up this problem and have this solution", were very rarely raised. I am unsure why there was a lack of initiative because, from what I could tell, people enjoyed working on the robot, so that was not the problem. I think that the most likely reason was a lack of a solid overall design for the robot. The construction of the robot was a piecemeal affair, which means that it was probably unclear exactly what work needed to be done. An effort was made at maintaining an online task list, but it was rarely used by anyone but myself and fell by the wayside after several months.

6.3.2 The Design Process

The initial design phase of the robot was far too short. A thorough list of the requirements for the robot was never made, which in turn meant careful consideration of the design choices was not made. Only a passing consideration was given to the design of previous tour-guide robots. As was mentioned in the discussion of the robot hardware, problems that cropped up were not thought through. The effect of this

short design phase was a robot that was essentially hacked together and lacked continuity in design. The lack of a clear design rationale made future design work difficult because there were no CAD files or other materials explaining why things were the way they were. Clearly, problems existed with the design methods used for the robot. The design process that I plan to follow for all future projects is explained below.

Projects generally start with a high-level idea, like Tourbot, a robot that will give tours of the MIT campus. From here, the idea needs to be fleshed out. With Tourbot, extensions on the basic idea were discussed but never really solidified. Given the evolution of the project as time has passed, a good design statement for the robot would be: the Campus Tourbot will be a robot that gives tours of various parts of the MIT campus. The robot will be completely autonomous, use ceiling-mounted fiducials for localization, and lidar for map-building and navigation. The robot will operate from a base station that it uses to charge its batteries. When giving a tour, the robot will use a combination of video, pre-recorded audio, generated speech, images, and data visualizations to provide a multimedia tour experience. While idle, the robot will interact with people using multiple modes. A face tracker will be connected to a servo-mounted camera to allow the robot to convey attention. Users will be able to ask questions to the robot by typing them, and the robot will respond as best it is able.

Following a detailed description of the overall project goals, each sub-area should be identified to allow for homing in on the needs for each piece of the overall system. On this front, the design breakdown for Tourbot was not too bad. The hardware could reasonably be split into chassis design, sensor selection, and computation. The electrical system had two needs: power distribution and automatic battery charging. The software design was much more complicated, as it covered a wide-swath of functionality. The breakdown that I arrived at was: communication, interfaces, localization, navigation, high-level planning, and human interactions. Each of these topics is covered in previous chapters.

From this high-level split, continue to recurse in each subsystem. At each level, identify the interfaces needed between the different modules. Each subsystem will

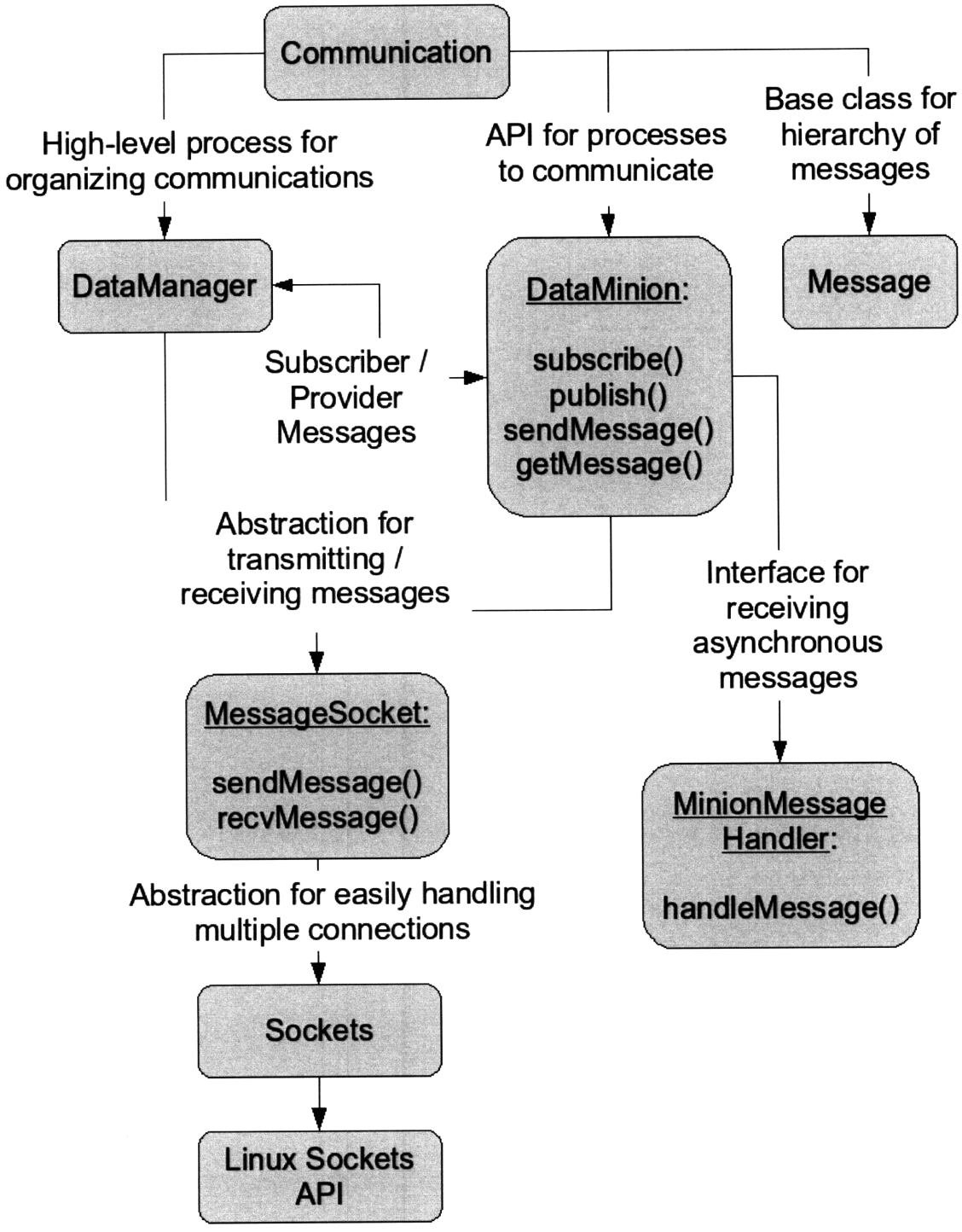


Figure 6-5: Major classes in the communication system and their purposes

have different number of layers, but the key is to making the abstractions and the interfaces. There will certainly be issues that arise while implementing that will make things need to change, but putting in the initial work here will really help speed development. The following example describes the design of the communication system (Figure 6-5).

The communication backbone for the robot was essential because every process running on the robot would need to communicate with at least one other process to function. At the highest level, I decided that processes would directly communicate with one another rather than through a single process because I wanted to avoid a bottleneck for message delivery. However, it would be cumbersome and inflexible for each process to need to know the exact location of every process that provided the type of data needed. To get around this requirement, I decided that a central process should handle all the routing needed to get connections established between subscribers and providers. Having a single process for this task meant that all running processes only needed to know the location of the central message process, the DataManager, in order to receive the data that they needed. After establishing this principle, I decided that the processes should be abstracted from the details of communication. All they should need to specify is the data they need and then that data should start arriving with no further effort. To accomplish this task, something would need to be embedded in each process that handled this task. The DataMinion was designed for this purpose. It allowed messages to be polled or delivered asynchronously and allowed messages to be sent to all subscribers with a single function call: sendMessage().

Both of the high-level pieces of the communication system, the DataManager and the DataMinion, relied on the ability to transmit and receive messages. The MessageSocket was designed to handle this task and deal with serializing the messages to allow them to be sent across the network. The MessageSocket allowed transmission of messages via the sendMessage() method and delivered messages asynchronously as they arrived. To function properly, the MessageSocket needed to be able to handle multiple connections. This ability was abstracted into the Sockets class, which han-

dled listening for new connections, establishing new connections, sending data, and receiving data. Each new connection was assigned an identifier that was used by the MessageSocket to send message data to the correct processes.

The abstractions and interfaces established in the communication proved to be robust and simple to use. Once they were implemented, everything just worked. Updates were only made when new messages were added to the system.

6.3.3 Building A Strong Foundation

Building a strong foundation and developing the core functionality for the robot needs to happen before any high-level functionality is considered. Why? The entire system depends on the low-level functionality for everything. When the core is well-built, more assumptions can be made at higher levels to simplify those problems, and there is one less place to search when debugging. If the core is not working well, then a lot of time will be spent floundering when this module stops working or that module stops working. For Tourbot, I identified the core functionality for the software to be the inter-process communication, low-level hardware interfaces, and basic odometry and driving.

When building a distributed system, the communications are the backbone. Each module needs to be able to receive data from other modules and to send processed output to other modules in order to function correctly. The communication system works well, and the ease of communication made the development of modules vastly simpler because any data was able to be reliably transferred to any other network location in the world with very few lines of code.

As mentioned before, each piece of hardware requries a special interface to abstract away the details of communicating with a device. The hardware interfaces on the robot worked well enough to gather/transmit data, but had quirks that were oftentimes annoying. The lidar interface would fail to properly initialize about half the time, requiring a power cycle of the lidar to get everything working. For over two years, the motor controller would crash when the E-Stop button was hit, requiring a hard reset of the device and re-running the interface, which would then reset the

robot odometry causing other potential issues. In all, the hardware interfaces were functional, but annoying and fragile. Instead of being methodical and getting functionality solidly working, I developed the interfaces until the functionality was enough to get the data, but not with the reliability that I should have given how important these interfaces are to the functionality of the robot.

The final piece of the robot's foundation is the motor control and odometry. A mobile robot has to move. When the robot is moving, it should have a guess as to its actual location. This part of the core was the biggest problem with the robot. For driving, I quickly hacked something together that enabled the robot to drive by arbitrarily converting desired metric velocities into actual commanded velocities. An effort was made to get a correct mapping between the abstract motor controller commands and the actual velocities, but it was never bullet-proof.

The robot's odometry was a source of constant pain. Between battles with an IMU and wheel slippage, the odometry was always a major source of error and frustration. When driving, the error that built-up was painfully obvious. The map would show a straight corridor as curved (see Figure 3.5.1), or the assigned goal of the robot would move from the center of the hallway to within the walls after only a few meters of travel. This error proprogated to all parts of the system and causes a number of nightmares in trying to work around the problem. However, about three weeks before the writing of this thesis (about 30 months into the project), I was moving some wires to a less exposed area and noticed that the mounts for the encoders were not rigid, but instead had some wiggle room that allowed encoders to move without the wheel moving. Clearly, this motion could cause the huge error in the odometry, so I took a hot glue gun and glued the encoders firmly into their mounts. Within moments, it became clear that the lack of a firm mount had been a major cause of the robot's poor odometry from the beginning.

6.3.4 Spiraling

A principle of engineering and design that has been emphatically preached to me since high school is K.I.S.S. (Keep It Simple, Stupid). Along with KISS, the use of

spiral development techniques, gradually adding functionality to a system, has been taught to me as the most effective method for building a complex system.

Despite the repeated emphasis on creating simple designs, I have a propensity for devising 'creative' solutions to problems and then forging ahead immediately without careful consideration of the alternatives. This head-first design method has yet to produce a solid and successful solution on the first try. Instead, I typically scrap the idea halfway through the implementation because too many details are unknown and the number of hacks exceeds the level of shameful code. At this point, I try to ground my ideas a bit more in theory from the literature and use the absolute *simplest* approach that *might* work.

For each subsystem, implement the simplest system that meets the bare functionality. For example, when controlling the motors, this approach means no velocity feedback, just set it and forget it. Doing the quick implementation allows integration testing to be performed at a very early stage, which is hugely beneficial because integration is where the vast majority of bugs crawl from their holes. Furthermore, integration testing will reveal the deficient parts of the system that require more sophisticated approaches.

A strong example of the spiral approach comes from the robot's path planning. My initial idea for the planner was to generate vector fields from all objects around the robot and then sum them to provide the way for the robot to move. This idea is not unprecedented and could have possibly worked, except I never thought about how I was actually going to determine the objects around the robot, and I did no math for determining how the vectors were to be generated. The net result was a couple thousand lines of code tossed by the wayside. I quickly switched from this approach to trying some sort of wall-following, relying on the lidar to find the current wall being followed. In locations where there was no wall, "fake" walls would be added to the map based on some map provided to the module. Once again, implementation revealed a number of big questions, and the code was all scrapped.

Following the failure of the wall-following with "virtual" walls, I decided to implement the most basic planner I could think of: an A* search through a grid. The result?

I had the robot successfully driving within two weeks and did not have to worry about the path planning while getting other pieces of the system up-and-running. After another year of work, the planner became the limiting factor, and I replaced it with system described in Section 3.5.

6.3.5 Using Math and Models

In the initial phases of software development, I would brainstorm ideas for how to solve a problem, determine a potential solution, and then forge ahead with design and implementation. I would consider the speed and running time of the algorithms being implemented, but did not consider any sort of mathematical foundation or justification for what I was doing. Developing in this fashion led to solutions that were inflexible and sewn together with kludges or hacks. In every case, a point was reached where the solution became a problem, and I was forced to scrap the prior work and try a completely new approach. An illustrative example of the advantages of math is the occupancy grid built by the robot.

The initial occupancy grid used on the robot was only loosely based on any published methods of building an occupancy grid. The environment around the robot started out as completely unoccupied. Then the raw lidar data would be placed on the map and expanded into configuration space. Each cell would become more occupied as it was observed over time. If a spot stopped being observed though, it would slowly fade away. This fade gave the robot a map memory of about 20 seconds.

The time-fade was a completely arbitrary calculation. I simply decided that the robot should forget the world after 20 seconds. A fading of the map can make sense if it is tied to another measured quantity, like the error accumulated in the odometry since the measurement was made. The time fade made planning extremely difficult because the robot would decide to navigate around an obstacle and begin moving. If the movement was hindered by the environment for some reason, the robot would forget about the obstacle it was trying to avoid and then drive right back through it. If the fade had been tied to the actual source of error in the robot, namely the odometry, then this problem would not have arisen.

The other half of the problm with this occupancy grid was the lack of a true sensor model. The grid was built by saying that a measured point is occupied, but a measured point is only a part of the total information given by a beam-based sensor. In addition to telling about the occupied space, the beam provides information on the unoccupied space. The line-of-flight nature of the lidar means that all space between the measured point and the robot must be free. This key insight, gleaned from [29], drove for the creation of the occupancy grid currently used by the robot. The use of a good sensor model and a mathematical basis for the map abstraction have made it much more useful.

Chapter 7

Future Work

This thesis has focused on my work in designing, implementing, and testing Alan Touring, a tour-guide robot. While working on this robot, a number of interesting questions have arisen that I was unable to investigate due to time constraints. This chapter discusses a number of areas that I would like to pursue in future research.

7.1 Robot Tourism

Tour-guide robots are nothing new. Over the course of the past decade, tour-guide robots ([2, 26, 31, 12, 30]) have emerged as an interesting and successful domain of service robotics. Robots guide visitors at museums world-wide.

Tele-operated robots have a long history, dating back to the 1950s [8], and have been used to find the *Titanic* [3] and explore other planets [1], among a wide variety of applications. In all of these situations though, the robot does the bidding of the human operator in a decidely asocial environment. The robot is being used to perform a task a human could not otherwise do.

In a series of papers [19, 20, 21], Eric Paulos and John Canny established a subdomain of tele-operated robotics call tele-embodiment. In tele-embodied robotics, the remote operator of the robot "becomes" the robot because they are able to communicate with people in the robot's environment using human modalities like speech, vision, and touch. The tele-embodiment scenario is interesting because an operator

is able to engage socially in a remote environment. The two-way communication is the distinguishing factor between regular tele-robotics and tele-embodiment.

Paulos and Canny's research focused on the creation of Personal Roving Presence (PRoP) devices that allowed people to become tele-embodied as a robot in a distant environment. The focus of PRoPs was acheiving human skills like communication without the need for a human form. The PRoPs they tested consisted of a camera, LCD, speaker, microphone, and pointer mounted on a post to a mobile base. These devices allowed a person to engage in two-way communications with people at a remote location. It was found that the remote users adapted readily to communicating using the PRoP, despite problems like network lag. The experiment subjects reacted positively to the experience indicating that they felt they were really a part of the remote environment.

Robot tourism is a tele-embodiment/tour-guide robot hybrid. A robot tourist is a semi-autonomous robot deployed at some interesting locale. The robot tourist is semi-autonomous because it handles the navigation and localization tasks, but the remote operator, or tele-tourist, assigns the locations that the robot visits. The robot has specific knowledge of its environment, like a normal tour-guide robot – in fact a robot tourist can also serve as a tour-guide. As it wanders through the world, the robot relays knowledge about the surroundings to the tele-tourist, providing a remote tour experience. Furthermore, the tele-tourist has control of the robot's communication system and is able to communicate with people around the robot. This capability provides true tele-embodiment and really allows the tele-tourist to experience the location by engaging socially with others, much as a tourist does when physically visiting an area.

Imagine such a robot deployed at the Eiffel Tower. While not being used remotely, the robot could interact with visitors to the tower, leading them around the grounds and talking about the tower's history. Once a remote user logs on though, the robot tourist does their bidding, most likely getting on the elevator and going to the top observation deck to really take in the surroundings. (This functionality might actually be useful for acrophobics visiting the Tower. Though perhaps they would still feel

the fear is tele-embodied; it would be an interesting topic to investigate).

A practical, and easier-to-implement, application of a robot tourist would be at a college campus. Many propspective students are unable to visit campuses when choosing a school due to lack of time or money. If a robot tourist was functioning at the school though, these students could remotely tour the campus and even attend an actual campus tour. The tele-tourist would be able to talk with students and administrators on the campus to ask questions and get a feel for the student body.

To my knowledge, no such robot tourist exists. RHINO [2] and Minerva [26] both had Web-based interfaces, but they were limited to selecting a tour for the robot to perform and then receiving the same tour information as local tourists. There was no two-way communication.

7.2 How Does Communication Affect the Perception of Robots?

This thesis contains a discussion on the general reaction of different segments of the public to the robot. These discussions were all informal though and happened while I was with the robot and clearly in control. I would like to perform controlled experiments to try and discover how communication affects a person's perception of a robot.

I have designed an experiment to try to determine the effects of communication on people's enjoyment of interactions with a tour-guide robot and how much information they learn during the tour. The experiment looks at the effects of one-way versus two-way communication during interactions with a robot.

The experiment scenario involves Alan Touring giving tours to guests at the MIT Museum in Cambridge, Massachusetts. On a tour, a group of museum guests follows the robot as it moves through the Robotics and Beyond exhibit on the second-floor of the museum while a remote user attends the tour across a network on the first-floor of the museum.

The exhibit features a number of robots constructed at MIT as well as displays and videos describing five major areas of robotics: artificial intelligence, sensing, moving, socializing, and learning and reasoning. At each of these displays, the robot stops and speaks about the topic. He gives a brief overview of the topic and then describes how the topic relates to his own capabilities. The descriptions are pre-typed text that is converted using a text-to-speech engine. Along with the descriptions are images, videos, and data visualizations generated by the robot. In between stops, the robot displays a visualization that provides an interpretation of what the robot is doing.

While the robot is leading a group through the exhibit, the remote user on the first-floor of the museum is also seeing and hearing the tour media played by the robot. The remote user receives an audio and video feed from the robot, which enables them to experience the robot's surroundings and to communicate with the tour group following the robot. A camera and microphone are also placed in front of the remote user so that their presence can be projected onto the robot.

Every other day of the experiment, communications between the remote user and the tour guests is one-way. The tour guests following the robot will see the remote user, but will not be able to have a conversation because the remote microphone will be turned off. To ensure that the same topics of conversation arise each day, a confederate will be assigned to either follow the robot as a tourist or to be the remote user. This confederate will have a script to follow.

At the end of the tour, each museum guest that attended the tour will be asked to fill out a questionnaire and take a short test based on the material presented by the robot.

7.3 Harassing Robots

As mentioned in Chapter 4.2, people tended to harass the robot in an attempt to make it fail. [2] and [26] also mention that people would try to force the robot out of its designated area. Furthermore, Tourbot would ask people to move if they were in its way, leading many people to wait in front of the robot until it asked them to

move, a result also mentioned in [2] and [26]. Doubtless, this behavior is common in many or all public robot installations. Why do people enjoy harassing robots? One reason may be that people are testing the robot's intelligence.

While the robot was at the museum, people were quite curious about how the robot moved through the world. I would discuss the robot's sensors and the map that it builds of the world. Once the robot started moving after explaining how the robot worked, these people would jump in front of the robot to confirm that the robot behaved as expected. During much earlier testing, the robot was swarmed by a a group of preschool students in the MIT Stata Center. The teacher jumped in front of the robot, proclaiming "Look! The robot won't hit me!". Immediately, it became the goal of the all the children to prove their teacher wrong. Both of these incidents seem to have the same cause: confirming the intelligence of the robot.

In the case of RHINO and Minerva, the robot was operating in isolation, so the people were most likely testing the abilities of the robot. There was no explanation (that I am aware of) of the capabilities of either RHINO or Minerva to museum visitors. Therefore, the visitors had some expectation or mental model of the robot's abilities before their encounters.

I wonder, then, what the general public's expectations are for a robot.

7.4 Curious Robots

A major limitation of Alan Touring is his reliance on a pre-built map and pre-loaded media and information about his environment. A much more useful information-based robot would be able to learn about its environment from experience and people and then relay that information to other people and robots. Such a robot would function as a focal point for information in its environment by agglomerating and distributing knowledge to inhabitants.

If a robot is operating in a human environment and is trying learn about this environment, it will need to gather information. I see three major categories of information important to the robot: physical and social. Physical information pertains to

the structure robot's environment, including the location of rooms, elevators, stairs, etc. and how they are interconnected. A robot with strong physical knowledge can answer questions like "Where is the nearest bathroom?" or "How do I get to 10-250?" much as a human would. The robot would give directions referencing landmarks and the topological layout of the environment, rather than specific distances and headings because they are more useful to a person finding their way. Social information encompasses knowledge about the society operating in the environment. For a college campus, social information would include questions like "What is the admissions process?" or "How many undergrads are there?" These questions could not be devised by the robot just by observing the physical world, but would need to be learned from people in the world. Another way differentiate the two classifications is: physical information would be useful for other robots, while social information is only useful for people.

Gathering physical information about the environment is crucial because without it, the robot will be unable to function because this information is used for the robot's navigation. Given the size and potential complexity of the robot's operating environment, the format in which the data is interpreted will have a great impact on its usefulness and accuracy.

A promising representation of the physical world is the spatial semantic hierarchy (SSH) developed by Ben Kuipers [15]. The SSH is a model for representing a large physical space that is inspired by the human cognitive map – the human representation of their environment – and consists of multiple layers of information linked by actions. Each layer defines its own set of objects and actions that relies on information provided by a lower level. The lowest level is the control level that defines the motor actions of the robot based on sensor readings and distinctive states are defined by some property of the raw sensor values. The next level of the hierarchy is the procedural level which defines views and actions. Views are the represented as the sensor readings at the distinctive states of the control level, and actions define the trajectories for moving between views. Following the procedural level is the topological level that defines places, paths, and regions. The places and paths create a

topological layout of the environment, while regions provide grouping. A robot can navigate through the map created at the topological level without any stored metric information about the world. The highest level of the hierarchy is the geometrical level where metric properties like distance, shape, and direction are attached to the paths and places in the topological level.

The construction and representation of the SSH is handled using an extended Voronoi graph (EVG) [4]. An EVG represents the world at the topological level. The basic representation of an EVG is the set of all point no more than distance M from a world feature. When building the EVG, a local perceptual map (LPM) is maintained of the region directly around the robot (an occupancy grid is one possible map representation), from the LPM, gateways and path fragments are extracted. A gateway is a place where the visible region of the world drastically changes, doorways and intersections are the most basic forms. Path fragments are the links between the gateways. Places in the world are built up from these gateways and path fragments. The exact definitions and algorithms for construction are much more involved and I refer the reader to [4] for a much more thorough description.

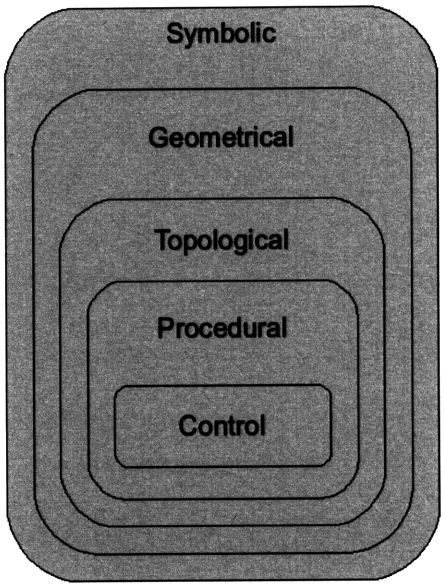


Figure 7-1: Layers of the spatial-semantic hierarchy. The symbolic layer is my proposed addition.

While the robot may be able to use an EVG to travel through the world, there is still a large disconnect between this representation and a useful human representation. I feel that an additional layer fits on the SSH: the symbolic layer. The symbolic layer adds a description of a place in a human language. Furthermore, it augments the geometrical description by tagging the metric information with better descriptions.

As an example, consider Building 4 on the MIT campus. This building is defined by intersections with Buildings 8, 10, and 12 as well as Buildings 2, 6, and 14. Each of these intersections is a four-way intersection, which is a good human-level description of the place. Additionally, Building 4 contains many rooms. These rooms would not be defined as actual places in the EVG, but are important pieces of physical information if the robot is going to provide directions. The SSH provides a means of integrating this information at the geometrical level, however, by defining the location of rooms based on their relative metric distance from a place or places. Some work has been done in this area with the use of a hybrid SSH in cooperative human-robot control of an autonomous wheelchair [5]; however, I feel that further refinements defined by the descriptive layer provide for a more descriptive map representation.

Representation of the social information contained in the robot is a more difficult task because it encompasses a greater variety of information, and many pieces of information can be completely unrelated to any other piece, or the relations between information is difficult and requires high-level reasoning about the content and context. Gathering this information is another challenge because it is so diverse and is not necessarily tied to any perception the robot can make. Given these difficulties, what approach should be taken for dealing with social information?

A primary source of learning social information would come from questions the robot is unable to answer. If the robot cannot answer, then it will post the question to an email list or online forum, where an answer can be given and added to the robot's knowledgebase. One problem with this approach is that the robot will only build up information very slowly. However, curiosity could be programmed into the robot to make it pro-actively seek information about its environment.

A curious robot would wander its world, remembering what it last saw. For physical features, like places and rooms, they can receive a label and not much more can be learned. With social information, the depth of knowledge is much greater. If the robot can recognize text in the world, that can provide a gateway into probing the depths of social knowledge. For example, if the robot sees a door labelled "Office of Admissions" and knows nothing of what 'office' or 'admissions' mean, it can ask a

person with whom it is interacting about these words. A question about admissions might lead to a discussion of students and deans and so on until the person tires of conversing with the robot and leaves some questions unasked.

The amount of information that could build up from these conversations is massive, so organizing it in a coherent and powerful way is a hugely important. One possibility is to simply store it as raw text and then perform searches through this body of text, though this approach seems crude and unrefined and leaves out valuable semantic information gained from the context of the conversation in which the information was received. It seems better to store the information in some hierarchy fashion similar to SSH, and likely derived from the function of the human brain. The exact format of this structure is a topic that would need to be researched much more.

7.5 Making Robots Act Naturally In Crowds

Despite considerable effort to make the motion of the robot natural and human-like, there were a number of quirks, like approaching people too closely or occasionally jerking into motion, that made it clear that the robot was, in fact, a robot. If we want robots to seem more human-like and be treated as social equals, they need to both verbal and non-verbal social skills. How can a robot be made to act more naturally?

The field of computer simulation of human environments has spent considerable effort in devising algorithms and methods for generating natural-looking human crowds [16]. [23] generates motions based on human psychology. In their research, they identify three classifications of collisions. Towards collisions occur when two people are walking towards each other. Away collisions occur when one person is behind another and is moving at a faster speed. A glancing collision occurs when two people are walking at about the same speed, but in slightly different directions and would bump shoulders.

To avoid a collision, one of three maneuvers is available (listed in order of human preference): changing direction, changing speed, and changing direction and speed.

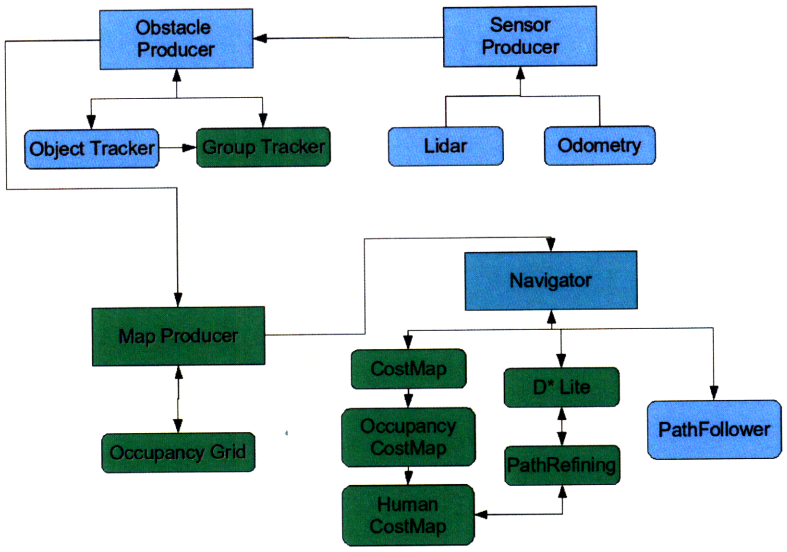


Figure 7-2: Flow chart for psychology-based robot motion

From observation human behavior, it was found that people try to minimize the amount of effort required to avoid a collision, and that the density of the people changes when an avoidance behavior begins.

If the crowd is too dense and traffic is moving in both direction, clusters of people form that are moving in the same direction. As the crowd density increases, the cluster sizes increase and people tend to join existing clusters because it is the easiest action to take to avoid collisions.

Given the above observations, a direct application of these principles to the path planning of a robot operating in a dynamic human environment seems promising, as the only information needed for the robot is the motion of people on the robot's map. To find a path that takes into account the movements of people would require a search in the state space of the robot that includes the time and velocity of the robot. Unfortunately, this state space is high-dimensional, so a search would be costly, though a sampling-based planning approach might yield good results.

An alternate solution is to use a decoupled planning approach (Figure 7-2). In this decoupled approach, an incremental search algorithm, like D*-Lite, would be used to find an initial path. After finding the path, potential collisions with moving objects are found by projecting the current velocity of the object onto the path of

the robot. Each collision is classified based into one of the previously mentioned categories. These collisions would then be converted into costs based on the desired avoidance strategy. The path would then be updated based on the new costs to determine the path that would actually be followed by the robot.

The other aspect of navigating in crowds is determining when the robot should join and move with a crowd and when it should overtake or breakaway from a crowd. [23] provides insights from psychology to help determine a solution to this problem. Observations show that people generally maintain their distance from one another until the density of people reaches a certain level. At this point, lanes of people heading in the same direction form, and people move in these lanes until their desired goal path deviates from the motion of the lane.

Using these guidelines, a robot could perform this common human behavior. A method for tracking groups of people would need to be devised. From this information, a planner would determine whether the intended destination of the group aligns with the robot's goal. If so, the robot would then join the group. To follow the group, a behavior-based system, similar to the system developed in [11], could be used.

A robot with the ability to navigate naturally amongst groups of people would combine the above two methods: avoiding collisions like a human and following groups. An algorithm would monitor the destination of the robot and switch between the planned navigation and behavior-based navigation depending on the characteristics of the people and objects around the robot.

7.6 Long-Term Deployment

Tourbot spent a month at the MIT Museum. During this time, he guided people through the Robots and Beyond under my close supervision and assistance. The time spent at the museum, however, was primarily for testing the robot in a small environment and gauging the reaction of the public to the robot's presence and design. The museum scenario is less interesting than a full, campus-wide deployment of the robot for a couple of reasons. First, there are museum tour-guide robots that

have more advanced capabilities and have been deployed for longer periods of time. Tourbot is not breaking new ground in the museum tour-guide world. Second, museum interactions are short-term interactions. The robot is around guests for 15-20 minutes and is speaking for the majority of this time.

A long-term deployment of the robot as a guide for parts of the MIT campus raises some interesting questions. As a resident of the MIT campus, Tourbot would encounter two distinct groups of people: short-term visitors to the campus and long-term members of the MIT community.

The group of short-term visitors consists of tourists and other temporary visitors to the campus, like conference attendees or prospective students. For this group, the interactions with the robot would be similar to a museum setting, short-term and mostly consisting of the robot providing information. However, the most common question that I am asked on the campus is "How do I get to Building X?", making the ability to answer this question essential for the robot to answer. Furthermore, prospective students have many questions about the campus that pertain to student life, academic programs, and many other things that would not be contained in a normal tour. To handle these questions, a database of facts about the MIT campus would need to be available for the robot to query. This database would allow the robot to try and answer any question it is asked by providing a few best matches. If none of these matches are sufficient, then the user could indicate this and provide an email address. The robot would then send an email to some mailing list that is established for answering questions. A member of the list would answer the question and it would be added to the robot's knowledgebase. The robot would then email the visitor the answer to their question. In this fashion, the robot would be able to build general knowledge about MIT and become more useful with time. An experiment to measure the increasing utility of the robot could be designed.

The group of community members consists of MIT faculty, administrators, students, and other employees. The people will all encounter the robot on a regular basis. At first, there is likely to be a novelty factor with the robot, as described in [13], where people investigate the capabilities of the robot and make their opinions.

Many community members need to find an obscure part of the campus at one time or another, which makes the direction-giving functionality useful. If that is the only part of the robot that is of use to community members though, then interest will quickly – and understandably – diminish. An interesting question to research would be "What makes a robot useful over a long period of time?" The Roboceptionist project attempted to maintain interest by having a robot with a backstory and personality that would change with time. While certainly a valid approach, the usefulness of this functionality is questionable.

One thing about the MIT campus is that there are many events happening every day that a very small subset of the community attend because publicity is weak to non-existent. A robot that is able to wander the campus and advertise for events might retain usefulness. Along these lines, I feel that access to a dynamic body of information and a changing set of capabilities is the key to long-term interest. Perhaps the robot could be used as an avatar for students on-campus. If one student is sick, maybe the robot could attend a lecture for them to watch remotely.

In any event, there are many factors to investigate with regards to the long-term usefulness of a service robot. The number of repeat users or first-time users could be tracked over time. Furthermore, the reasons that people are using the robot would be interesting to watch. If a new capability is added to the robot, how do people find out? The usefulness of a feature could be found by measuring the number of users of the feature over time. If a feature has a very high initial use, but then quickly tapers off, it probably was not very useful. However, if a feature sees high initial use or a slow buildup to some steady-state, that would be more indicative of the type of feature that is useful for an information-based service robot like a tour-guide.

Chapter 8

Conclusions

This thesis has described the design and implementation of a tour-guide robot, Alan Touring, the Campus Tourbot. The robot was designed to be an information-based social robot that would wander parts of the MIT campus and provide tours to guests and answer questions about its environment. Design of the robot began in October 2005 and full development started at the beginning of 2006. During the past 31 months, I have worked extensively on the robot, dedicating thousands of hours to its design and implementation.

Going into the project, I had some experience building robotic systems, but had never built anything from the ground up with anywhere near the complexity and ambition of the Tourbot project. As a result, I was learning as I worked. Building the robot in this fashion meant that many avenues that I explored were dead-ends, causing months of work to be scrapped in favor of alternate approaches which led to the protracted and incomplete development. Through the tumultuous development of the robot, I gained the invaluable experience of working on a real system of my own design in real-world conditions, which differ extensively from even design-based courses. At the end of my time developing the robot, the final goal of the project has yet to be reached. Important milestones for attaining a truly autonomous robot have been reached, but a number of pieces remain incomplete or broken, keeping the robot from its destiny.

The software architecture developed for the robot uses an asynchronous, message-

passing system based around anonymous publish/subscribe. This architecture is very well-suited to this robot because there are a vast array of processes that need to run for the robot to function. If a more constrained architecture had been used, the complexity of the system would have exploded beyond containment. Currently, when the robot is fully-functioning, 22 processes are operating simultaneously. All interprocess communication is handled efficiently by the DataManager/DataMinion system that I wrote.

At the current time, the robot is able to successfully navigate amongst people and complete tours of the MIT Museum with little human intervention. It is likely that robot could operate in the Infinite Corridor on the MIT campus as well, though this capability has not been tested due to the effort needed to place fiducials all across the campus and the lack of manpower to do so.

To successfully give a tour, the robot relies on robust fiducial detection, short-term dead reckoning, occupancy grid mapping, an incremental path planner, and a motion controller capable of following the discovered paths. Each of these pieces works very well independently and when integrated into the system as a whole.

The fiducial detection is used to keep the robot oriented in a global coordinate frame and to provide waypoints to follow while giving a tour of its environment. The detector uses a combination of dynamic threshold adjustment, binary segmentation, and a number of constraints on the fiducial size and shape to robustly find fiducials in images captured by the robot's ceiling-facing camera. Very few false positives are found, and most of these are caused by distortion from the camera lens that I have been unable to properly correct.

The occupancy grid mapping performs well and provides a consistent map to use for path planning. Standard occupancy grid mapping using a single lidar or lidars with non-overlapping fields-of-view is simple to implement. The challenge posed by Tourbot's lidars was caused by fields-of-view that overlapped in the x-y plane, but not along the z-axis. The standard implementation causes the lidars to contradict one another and erase objects from the map that were seen by only one of the lidars. To solve this problem, I transitioned the occupancy grid into three dimensions. To reduce

the space needed for a three-dimensional grid, I maintain only the maximum observed height of an obstacle at each position. By adding only this value, the size of the grid only doubles rather than growing by a factor of 12 if a consistent discretization was used. Furthermore, the complexity of the search is reduced because the occupancy values of the z-axis are not maintained independently, so they do not have to be merged and projected into the x-y plane for path planning.

Path planning on the robot is handled by the incremental D*-Lite algorithm. D*-Lite performs an initial search to the goal and then reuses much of the previous search information to reduce the space that it searched when a new plan is updated. The searches occur in the configuration space of the robot, found by marking as occupied all points within 0.6m of an occupied position in the occupancy grid. The occupied positions have an additional falloff that keeps the robot from travelling too close to walls and static objects unless absolutely necessary to make progress through the world.

Once paths are found, motor velocities are generated by a controller that relies on a constantly moving goal to smoothly move the robot through the world. The idea behind the controller is that the robot should drive at maximum speed when it needs to travel straight and should slow down as it turns. The controller works well for environments where the robot does not need to track the discovered path with extreme accuracy and smooth motion is desired. The lack of tracking accuracy is assuaged by the incremental planner that is efficiently updating the robot's path, which lessens the effect of deviating from the desired path.

I mention above that the robot is able to complete tours with little human intervention, meaning the robot cannot operate in 100% autonomous fashion. The reason that the robot cannot operate without supervision and occasional help is caused by objects that are invisible to the robot and no current method for the robot to safely deal with them. Glass is invisible to the robot because lidar sees straight through it. The robot relies solely on lidar for obstacle detection, so the invisibility of glass is a major problem. Discussions and designs for a bump skirt that would give the robot tactile feedback from the world have been in the works for about 18 months,

but have yet to come to fruition. A bump skirt is still a poor solution though, as the robot weighs almost 300 pounds and has considerable inertia, so a collision with a glass wall or display case could cause considerable damage.

The root of this problem was the decision to give the robot only a relative map of fiducial locations and rely purely on sensors for building a map and finding obstacles. This choice was made to keep the robot flexible in the environments in which it could be deployed because fiducials could be placed without needing to also build an accurate metric map. A solution to the problem would be to augment the current fiducial maps with untraversable regions, and then ensure that a fiducial is placed next to every forbidden area. This approach is similar to that used by RHINO [2] and Minerva [26], where invisible objects were added as imaginary sensor readings in order to place them on the map. This approach would also solve the invisible stairs problem which is the biggest threat to the safety of the robot.

As stated previously, it is clear that the robot is on the cusp of complete autonomy in its environment. However, this progress occurred at the expense of development of the social capabilities of the robot. At the present, a simple touchscreen interface allows people to start and pause a tour, as well as learn a bit about the history of the robot. Beyond this capability though, no interaction is possible. When giving tours, the robot is fully able to play all forms of media, but it can only output, no input can happen. For a robot that claims to be social, this absence is a huge flaw.

The robot needs a way to maintain a dialogue with users. One of a tour-guide's primary responsibilities is answering questions that guests pose. Providing the robot with a querying system that is dynamic and expandable is the major step needed for the robot to attain its goal as a functioning tour-guide. The robot is not expected to know every answer, but it should be capable of finding the correct answer and then remembering it for the future. An alternate solution is to have a tour-guide accompany the robot on its tours to field questions from the guests, but this solution pushes the robot from the realm of a social robot and makes it a one-way media device.

Robots are going to begin playing a more important role in society as they become

more robust and flexible in the tasks they can handle. My experiences interacting with the general public and Tourbot shows that there are definite expectations for the capabilities of robots like drawn from many aspects of society. People expect robots to be able to understand spoken language and react to all manner stimuli in the environment. Unfortunately, the current state of speech recognition denies robots this ability. If social robots are going to become integrated into society, they will need to understand what people are saying.

Though not entirely successful, my work on Alan Touring has provided me with many interesting ideas for future robots, like robot tourism and robots that are curious about their environment, learn about their environment, and then share what they learn with people and other robots. Imagine a museum where dozens of robots roam the exhibits, asking questions of the visitors, answering visitors questions, and then talking with their fellow robots about what they learned. Beautiful. I see a bright future information-sharing robots.

Bibliography

[1] Jpl mars pathfinder. `http://mars.jpl.nasa.gov/MPF/index1.html`.

[2] Wolfram Burgard A, Armin B. Cremers A, Dieter Fox B, and Dirkhhnel A. Experiences with an interactive museum tour-guide robot, June 1998.

[3] R.D. Ballard. A last long look at titanic. *National Geographic*, December 1986.

[4] Patrick Beeson, Nicholas K. Jong, and Benjamin Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *IEEE International Conference on Robotics and Automation (ICRA?05)*, pages 4373–4379, 2005.

[5] Patrick Beeson, Matt Macmahon, Joseph Modayil, Aniket Murarka, Benjamin Kuipers, and Brian Stankiewicz. Integrating multiple representations of spatial knowledge for mapping, navigation, and communication. In *Proceedings of the AAAI 2007 Spring Symposium on Control Mechanisms for Spatial Knowledge Processing in Cognitive / Intelligent Systems*, 2007.

[6] Pantelis Elinas, Robert Sim, and James J. Little. σslam: Stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Proceedings of the IEEE Conference on Robotics and Automation*, Orlando, FL, 2006.

[7] Jodi Forlizzi and Carl DiSalvo. Service robots in the domestic environment: A study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 258–265, 2006.

[8] Raymond Goertz and R. Thompson. Electronically controlled manipulator. *Nucleonics*, 1954.

[9] C. Havasi, R. Speer, and J. Alonso. Conceptnet 3: A flexible, multilingual semantic network for common sense knowledge. In *Proceedings of Recent Advances in Natural Languages Processing*, 2007.

[10] Berthold Klaus Paul Horn. *Robot Vision*. The MIT Press, 1986.

[11] Hiroshi Ishiguro, Takayuki K, Takahiro Miyashita, and Henrik I. Christensen. Navigation for human-robot interaction tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1894–1900, 2004.

[12] Gunhee Kim, Woojin Chung, Kyung-Rock Kim, Munsang Kim, Sangmok Han, and R.H. Shinn. The autonomous tour-guide robot jinny. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 4:3450–3455 vol.4, Sept.-2 Oct. 2004.

[13] Rachel Kirby, Frank Broz, Jodi Forlizzi, Marek Piotr Michalowski, Anne Mundell, Stephanie Rosenthal, Brennan Peter Sellner, Reid Simmons, Kevin Snipes, Alan Schultz, and Jue Wang. Designing robots for long-term social interaction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 2199 – 2204. IEEE, August 2005.

[14] S. Koenig and M. Likhachev. D* lite. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 476–483, 2002.

[15] Benjamin Kuipers, Rob Browning, Bill Gribble, Mike Hewett, and Emilio Remolina. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[16] Richard Leggett. Real-time crowd simulation:a review. December 2004.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[18] James McLurkin. Swarm research. `http://people.csail.mit.edu/jamesm/swarm.php`.

[19] Eric Paulos and John Canny. Delivering real reality to the world wide web via telerobotics. In *In Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1694–1699, 1996.

[20] Eric Paulos and John Canny. Designing personal teleembodiment. In *In IEEE International Conference on Robotics and Automation (ICRA*, pages 3173–3178, 1999.

[21] Eric Paulos and John Canny. Social tele-embodiment: Understanding presence. *Autonomous Robots*, 11(1):87–95, 2001.

[22] Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation – mobile robot navigation with uncertainty in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Detroit, MI, May 1999.

[23] Stephen J. Rymill and Neil A. Dodgson. A psychologically-based simulation of human behaviour. In *Theory and Practice of Computer Graphics*, Canterbury, Kent, UK, June 2005.

[24] Jamieson Schulte, Charles Rosenberg, and Sebastian Thrun. Spontaneous, short-term interaction with mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 658–663, 1999.

[25] Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21:735–758, 2002.

[26] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, and C. Rosenberg. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research*, 19:972–999, 2000.

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

[28] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent map acquisition and localization for mobile robots. In *Machine Learning*, pages 29–53, 1998.

[29] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[30] Nicola Tomatis, Roland Philippsen, Björn Jensen, Kai O. Arras, Gregoire Terrien, Ralph Piguet, and Roland Siegwart. Building a fully autonomous tour guide robot: Where academic research meets industry. In *Proceedings of the 33rd International Symposium on Robotics (ISR'02)*, Stockholm, Sweden, 2002.

[31] Thomas Willeke, Clay Kunz, and Illah Nourbakhsh. The history of the mobot museum robot series: An evolutionary study. In *Proceedings of FLAIRS 2001*, 2001.