

Table of Contents: Implementation and Evaluation of a Collision-Avoidance Navigational Algorithm on a Mobile Robot

PREFACE	II
GLOSSARY	IV
ABSTRACT	V
CHAPTER I: INTRODUCTION	1
1. THESIS STATEMENT.....	1
2. PROBLEM DEFINITION.....	2
3. REVIEW OF RELEVANT LITERATURE.....	3
4. RATIONALE.....	4
5. IMPACT STATEMENT.....	5
6. OVERVIEW OF THESIS REPORT.....	7
CHAPTER II: TRANSLATING THE ALGORITHM INTO SPECIFICATIONS	8
1. GENERATING ENVIRONMENTAL REPRESENTATION.....	8
2. REDUCING THE DATA.....	9
3. DETERMINING STEERING DIRECTION.....	9
CHAPTER III: CODE DEVELOPMENT	11
1. DETERMINING EXTERNAL INPUTS AND OUTPUTS.....	11
2. CONTROL STRUCTURE.....	12
3. TESTING.....	12
CHAPTER IV: INTEGRATION	14
1. COORDINATING INDEPENDENT SECTIONS TO WORK IN CONJUNCTION.....	14
2. TESTING INTEGRATED CODE.....	15
3. IMPROVING PERFORMANCE.....	15
CHAPTER V: SIMULATION	16
1. TESTING.....	16
2. FINDING BUGS.....	16
3. REMOVING BUGS.....	18
4. REMAINING PROBLEMS.....	18
CHAPTER VI: MARCUS	21
CHAPTER VII: EVALUATION AND CONCLUSIONS	22
1. INTERPRETING THE VFH DEFINITION.....	22
2. USING MARCUS' FACILITIES.....	22
3. CONCLUSIONS.....	23
4. RECOMMENDATIONS AND CONTINUING EFFORTS.....	25
BIBLIOGRAPHY	26
APPENDIX A: FIGURES	30
FIGURE 1: MAPPING ACTIVE WINDOW CELLS ONTO THE POLAR HISTOGRAM.....	30
FIGURE 2: SELECTING A SAFE DIRECTION OF MOTION BASED ON OBSTACLE DENSITY.....	31
FIGURE 3: SCREEN-SHOT OF THE SIMULATOR.....	32
APPENDIX B: MARCUS, THE MOBILE ROBOT	33
FIGURE 4: PICTURE OF MARCUS.....	33
STATISTICS FOR MARCUS.....	33

Preface

I have been interested in the field of robotics for as long as I can remember. Some of this interest is rooted in science fiction depictions of futuristic autonomous robots. Particularly inspiring have been the literature of Isaac Asimov and Hollywood films such as the Terminator, the Star Wars trilogy, and 2001. All of these fictional works outlined possible consequences of creating autonomous, intelligent, and capable robots.

Some of these images depicted the robots of our dreams; these robots were intelligent, yet they remained subservient to humans. Other images shown in these sci-fi classics present nightmarish visions of the destruction that robots could have on humanity, overthrowing people. Most impressionable was the image of HAL from 2001. HAL had a strong sense of self-preservation. When he feared that the humans on board his ship were planning to shut him off, he tried to kill all the crew-members, succeeding with all but one.

Over last summer, while I was working at NASA's Goddard Space Flight Center, I became caught up in the excitement of the Mars Pathfinder. Touching down on the surface of Mars on July 4th, 1997, the Mars Pathfinder successfully launched a teleoperated robotic rover onto the surface. This robot, the Sojourner, transmitted an enormous amount of visual data of the Martian surface, which has never before been seen by human eyes, back to Earth. I knew then that I wanted to get involved in the field of robotics at some point in the future.

One small, yet vital, part of the field of robotics is in creating artificial vision. Without a visual system, a robot can not perceive its surroundings, interact with other

objects on its own, or navigate safely and effectively. Artificial vision is a captivating field because it is still in its infancy. There is much that needs to be resolved to determine how a visual system should work. Yet, it is common to look to natural visual systems for clues on how to approach designing vision for robots. In this way, robot vision work is reverse-engineered from the efficient and effective working examples in nature.

Computer vision is an exciting field, one which requires much effort in the design phase. I particularly enjoy the design phase of any project best of all, and I am very excited about the day when robots will be thoroughly commonplace. For these reasons, I was inspired to get involved in research with the UVA Vision Group within the computer science department.

Glossary

algorithm	precise method serving as a solution to a computer problem
candidate valley	direction that is safe for the robot to move in
Cartesian grid	grid representing the entire world of the robot; the values in the cells represent the obstacles in the room
histogram	frequency distribution
Marcus	UVA Vision Group robot
polar histogram	histogram of the region near the robot relative to the robot's position
real-time	acting quickly upon any current situation
sector	one slice of the polar histogram containing one value, the obstacle density for that sector
Vector Field Histogram (VFH)	a fast, obstacle-avoidance navigational algorithm for mobile robots

Abstract

For my thesis, I implemented and evaluated the VFH fast obstacle-avoidance navigational algorithm on UVA's mobile robot, Marcus. To do this, I worked with another student, Matthew Davidson, who wrote the part of the program that creates a Cartesian grid of the environment about the robot and then smoothes the grid into a polar histogram. I wrote the part of the program that used the smoothed polar histogram to determine a steering direction for Marcus. Also, I wrote the control for the entire process.

First, we interpreted the ambiguous description of the VFH algorithm into specifications. From the specifications, we decided exactly how to break up the program and decided upon data structures to use throughout the program. Once we decided how the program would communicate, we wrote and tested our parts separately.

Integration involved resolving unforeseen conflicts. When the program was unified, we tested it on the robot simulator to avoid destroying the robot with unanticipated glitches. During this simulation phase, we determined that there are many inherent flaws with the world representation used with this algorithm. Carefully monitoring each run, we tested our implementation directly on Marcus.

The final step to my thesis was the evaluation phase. The first result was that the definition for VFH was ambiguous, which caused some problems in the design and implementation phases. Also, the corporate literature pertaining to Marcus and its simulator was misleading and unsuitable in some cases, making it very hard to work with. The algorithm worked on the simulator in some cases; however, inherent problems with the VFH approach prevented success in many other cases.

Chapter I: INTRODUCTION

1. Thesis Statement

For my thesis, I implemented Vector Field Histogram (VFH), as defined in the paper written in 1991 by Borenstein and Koren, on the UVA Vision Group's Mobile Robot, Marcus. Vector Field Histogram is a fast obstacle avoidance navigational algorithm. VFH uses sonar to detect obstacles. The data is then reduced and analyzed to determine a safe and appropriate direction in which to steer. My work was done in conjunction with another student, Matthew Davidson, who inputted echoed sonar data and interpreted it into a histogram, representing the probabilities of an obstacle in a certain location with respect to the robot. My particular part of this implementation was to take this histogram and analyze it for possible safe ways to navigate. Then I implemented the decision-making rules that choose an appropriate direction in which the robot should steer.

The way I did this was by following the first few software life-cycle steps. I interpreted the algorithm into specifications for a design, designed the flow of the program and picked the data structures to use, implemented the design into code, and tested the results. Finally, I iterated through this process. In order to reduce risks, I made use of throw-away and evolutionary prototypes while implementing and designing.

My work in implementing this design provides groundwork that more research in this field will use. A few years down the road, a reliable collision avoidance navigational algorithm will be developed, sparking the dawn of mobile robots.

2. Problem Definition

The Objective

The overall objective was to implement and then evaluate the Vector Field Histogram (VFH) algorithm on the UVA Vision Group's robot, Marcus. This objective can be broken into subgoals. The precise definition of the VFH algorithm, as defined in “The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots” by Borenstein and Koren (13), can be used to define the program specifications. These specifications lead to the determination of data structures appropriate for storing navigational information. Data structures can be turned into programming language syntax to represent the components of the algorithm. To test and debug the program, the robot navigation simulator can be used. The program can also be uploaded onboard the robot and tested further.

The algorithm

Obstacles are observed using sonar, which is a technique of sending sound waves out and using the echoing sound waves to determine where objects are located in relation to the source. The information that is returned by the sonar is translated into a 2-dimensional grid of the environment around the robot and stored as the histogram (14,

280). After the sensors read data that an obstacle is present, one cell in the region of the histogram that represents the location of the obstacle is incremented (14, 280). By continuously and rapidly scanning for obstacles, the robot builds a grid to represent its environment using a probabilistic distribution (14, 280). Two stages of data reduction then are employed (14, 281). The data reduction essentially maps the active region of the histogram onto a polar histogram, which is a histogram relative to the robot's position. See Figure 1 of Appendix A for a visual interpretation of this process.

Once the data is simplified to represent a smooth polar histogram with respect to the robot's position, the algorithm examines the new histogram to determine candidate valleys, regions for safe robot movement (14, 283). The algorithm chooses the candidate valley that is closest to the direction in which it wants to move. See Figure 2 of Appendix A for a picture of the candidate valley assessment process.

3. *Review of Relevant Literature*

There are a handful of sources that I used most heavily during the project. In particular, the document that defines the Vector Field Histogram (14, 278-288) was particularly useful. It steps through the algorithm methodically, providing the understanding of the implementation specifications.

In order to implement anything on Marcus, I needed to understand how to use the features of the robot. To learn about the robot, I relied upon the three robot reference manuals. The reference manual that pertains to the hardware of the robot (16) provided

details about the robot's hardware. By using it, I understood the physical layout and components of the mobile robot, Marcus. The language reference manual (17) provided language rules appropriate for manipulation of the robot. It explained semantic and syntax rules for functions that could be used to manipulate the robot. Finally, the user's manual (18) described the possible ways to use the robot. It helped me learn about the available ways to manipulate the robot.

4. Rationale

Robotics is a field that changes everyone's lives. Robots will make people's lives easier and safer. Institutions and corporations all over the world are researching and developing new venues for robots. Applications for robots will soon be far more widespread than they already are. In some ways, robots will help us, and in some ways they will replace us. Regardless, robots are stepping into our lives more. Our world will never be the same (1).

There are many ways in which robots already do the routine and dangerous jobs that people don't want to do. Robots will continue to accomplish these tasks. The spectrum of applications for robots will increase dramatically over the next several years, as robots continue to replace human involvement in three areas of work-dirty, dull, and dangerous work. In order to make these applications for robots work, robots need to come equipped with a visual system that will allow them to move autonomously. The visual system has to be reliable and fast, providing the robot with factual information about the terrain about it. Otherwise, the robot would be unable to avoid obstacles.

Consequently, unavoidable crashes will occur, damaging expensive equipment and injuring natural life-forms.

That is where my project comes in. The Vector Field Histogram is a fast obstacle avoidance navigational method for mobile robots that uses a histogram to represent obstacles in the world around it. These obstacles are observed using sonar. The algorithm then tells the robot how to navigate to avoid these obstacles, and it does this in real-time. The advantages of Vector Field Histogram over other navigational algorithms for mobile robots are that VFH allows fast, continuous, and smooth motion. By implementing and studying the effects of this algorithm, we are expanding the field of collision-free mobility for robots. We are equipping the field of robotics with the eventual ability to provide autonomous applications for mobile robots. This ability will provide robots with the capability to work and move on their own.

5. *Impact Statement*

My particular project of implementing a fast navigational technique for the UVA robot, Marcus, will be used as part of ongoing engineering research into mobile robot collision avoidance. The critical path to the day of ubiquitous mobile robots is in generating a consistently accurate algorithm for collision avoidance during mobility. My research and enhancements on my work will lead to a time when a reliable collision avoidance algorithm will be accepted and integrated into numerous mobile robots.

There are already many applications in which robots are used. Manufacturing is an area that currently employs many robots. Robots are being used to provide quick, reliable work and quality control for many factories. Another modern application of robotics is to perform activities that are unsafe for people. Militaries use special robots to traverse possible landmines to detonate them without shedding human blood (1). Likewise, police use special robots to inspect, remove, and safely detonate bombs (1). Robots were employed to clean up the disaster of 3-mile island without further damage to humans (1). Robots are used to clean airplane wreckage if there was a nuclear missile aboard. They have also been used to walk into active volcanoes, providing data that people could not otherwise collect (1). In addition, robots have been employed to report upon the conditions on other planets, places currently unfeasible for human exploration.

When a collision-free navigational system is perfected, mobile robots will become ubiquitous throughout industry and life. These robots will perform routine tasks involving little more than collision-free motion, replacing the need for many human workers. For instance, these robots will be in the form of automated vacuum cleaners and mops, deliverers of medical and food supplies within hospitals, and self-propelled lawn-mowers and tractors. These applications will free up time and expenses for companies, but those put out of work will resent them. Another application for these self-driving robots is navigation in and manipulation of environments that are unsafe or unfeasible for human participation. For instance, robots will be used more extensively to clean up nuclear waste in contaminated regions. In addition, they will be more heavily used for planetary probes. Reducing the risk to human lives, these robots could explore regions that prove unsuitable for humans.

6. *Overview of Thesis Report*

The first chapter of this technical report introduces my thesis. It describes the rationale and the impacts of my thesis project. The second chapter shows how I translated the algorithm into specifications. Discussion of the code takes place in chapter three. The fourth chapter describes the integration of my part of the algorithm with the part that was done by the other student. Running the simulator is described in chapter five. The sixth chapter describes uploading the program onto the robot. The final chapter gives an evaluation of this navigational algorithm and concludes.

Chapter II: Translating the Algorithm into Specifications

As outlined in the paper by Borenstein and Koren, there are three overall tasks involved in the VFH algorithm. The world representation must be created. Then that representation needs to be reduced. Finally, a single steering direction needs to be chosen (14, 281). In this chapter, I discuss the way in which I took the ambiguous natural-language description of VFH tasks and converted them into rigorous specifications for my program.

1. *Generating Environmental Representation*

The first part of the VFH algorithm is to directly translate the sonar data into a representation of the entire environment around the robot. The robot sends off the sonar. The sonar comes back with the distance it traveled before hitting an object. This information is used to determine which cell to update in the grid that represents the robot's world. The angle that the robot is facing plus the angle of a particular sonar sensor represents the direction of the grid cell to update. The distance to the nearest object in that direction points to the location of the cell to update. In order to update the cell, the integer value in that cell is incremented. By continuously scanning and updating this grid, the robot generates a world environment representation (14, 281). This world environment representation is also known as a Cartesian grid.

2. *Reducing the Data*

The second part of the algorithm is the first data reduction. It takes the world coordinate system that was computed in the first part of the algorithm and reduces the information (14, 281). It does this by changing the Cartesian representation, which has data values in every cell, into a polar representation that contains one value per sector (14, 281). In this way, the algorithm sums all the cells in one region of the world coordinate system. That sum value becomes the value for one sector of the polar histogram, and it represents the obstacle density for that region (see Figure 1). This polar representation depicts the active region directly surrounding the robot as a compressed version of the world representation.

3. *Determining Steering Direction*

The third part of the VFH algorithm takes the sector information and reduces it into a single steering direction (14, 281). The first step is to figure out the target direction. The target direction is computed using the coordinates for the target and the current robot location. Once the target direction is determined, it then is translated into the target sector. The way in which this conversion is done is simply by figuring which sector of the polar representation contains the target direction. This sector becomes the target sector. Starting with that target sector, the algorithm starts looking to the right and left until it finds the first sector to satisfy a certain condition. This condition is that the value within that sector must be below a certain threshold. This sector is the sector

nearest to the target in which the robot can maneuver without crashing, and it represents one edge of the candidate valley (14, 281).

The algorithm then continues in that direction until it finds the last surrounding sector that is below this threshold, the other edge of the candidate valley (14, 282). This sector represents the other edge of the region that is safe to maneuver within. If the number of sectors in this safe-to-maneuver region exceeds some maximum value, then the robot considers the candidate valley as only that maximum number of sectors wide (see Figure 2).

Chapter III: Code Development

In order to change the specifications into program structure and design, some issues must be decided. First of all, you need to decide whether to split the program up into separate parts, for separate individuals. If the work is split into parts, like it was in our case, then there is a whole extra layer of work that must be done in the design phase. In this case, the external inputs and outputs for each part need to be decided up front. Also, the overall data structures and the ways they will interact need to be determined early. An additional thing that needs to be determined when designing the program is the control flow of the program. The design should be broken into functional components and the calling order should be decided. The process of designing the program structure is explained in this chapter.

1. *Determining External Inputs and Outputs*

The first step to writing the code was to break up the program into parts for both me and Matthew Davidson, the other person working on this project. After deciding the way to split up the project, we discussed the inputs that I would need from his program and the inputs that he would need from the robot. We decided that he would need to be able to get the robot's coordinates from the robot. My part also needed the robot coordinates. In addition, his section also required the returned sonar values in order to generate the world histogram.

2. Control Structure

The next step to structuring the program was for us to decide how the program should function and flow together. We decided how to group certain functional components and how to let those components communicate with one another.

We decided that his part would have several classes, user-defined data types that would contain and structure information specific to a particular aspect of his part of the program. We also decided that my part would control the flow of the program. It would call his part to get updated representations of the world around the robot, and then it would use those world depictions to determine an appropriate steering direction.

3. Testing

This stage of testing involved individually smoothing out bugs in our parts of the programs. We worked out compile time errors, and we also ran functional tests to determine if the program did what we expected and wanted it to do.

The tests that I ran on my part of the program included testing to see if the target direction was calculated correctly, given values for the current robot coordinates and the target coordinates. Once I established that the target sector value was computed correctly, I tested the way in which my part of the program recognized the bounds for a region in which it could navigate.

There were many cases that needed to be tried in order to determine that the algorithm properly discovered the boundary of a safe region for navigation. The cases included checking what my program did in case there were no appropriate regions to

navigate in, in case there were multiple safe regions, and in case there was only one safe region. In the cases where there was at least one safe region to navigate in, I tested to see if my program found the proper edges. I checked to see if my program worked properly when the candidate valley contained the lowest and the highest sector numbers, which happen to be right next to each other since these sectors are situated around a circle. In other words, if there are 72 sectors in the polar histogram, then one case that I checked was one that contained both the sector numbered 0 and the sector numbered 71. Once we both individually completed testing our parts, Matt and I were ready to integrate the two parts together.

Chapter IV: Integration

In the integration phase of this project, we had to smooth our parts of the program together. Some parts of this process were straightforward control issues. Other aspects to this phase were not so straightforward; they involved resolving the differences between opposing interpretations of details we overlooked in the design phase. Once these issues were taken care of, we worked on testing the unified program and improving its performance, where possible. The entire integration phase is described within this chapter.

1. *Coordinating Independent Sections to Work in Conjunction*

In this part, I fit his code to mine. His code, which generated a world coordinate system and did the first round of data reduction had a separate control function that needed to be combined with the control function in my half of the program. I had to develop an appropriate function that would serve as the control loop for the entire program, retrieving the information that his part generated about the robot's world and analyzing that information to find a steering direction.

There were a few unexpected difficulties in getting our two halves to work with one another smoothly. These difficulties were related to duplicated efforts on our parts, to unforeseen details, and to aspects that we interpreted and approached in different, incompatible ways. For instance, we both had created separate constant variables that represented the same thing and should each have only been defined once. Another type of problem that we stumbled across involved the details of the program that we failed to

formally specify together. There were a couple of instances where we interpreted the program requirements differently or had a different image of what should ideally happen.

Luckily, due to our structural preplanning, there were remarkably few of these integration problems.

2. *Testing Integrated Code*

Once the two parts of the program seemed to fit together well, I had to try testing to see if they worked together. The majority of this work was done in compiling and linking these two halves together. Once there were no compile-time errors, we were ready to move on to the real testing, which was done on the simulator.

3. *Improving Performance*

In order to improve performance, we moved some of the steps that were unnecessarily being repeated every time the robot received and processed sonar information outside of the main control loop, which is called continuously and often.

Chapter V: Simulation

Testing of our unified program was accomplished on the simulator. We used the simulator to remove errors before importing the code onto the robot. If we had omitted this step, we might have inadvertently commanded the robot to drive into obstacles. We viewed this simulation step as essential, because it gave us the chance to get the problems out of our code before uploading it onto the robot. The simulation process is described in further detail within this chapter.

1. *Testing*

The majority of the testing that we did for this program was done during the simulation phase. This is because the simulator graphically showed us where we were running into problems with our code and where our code worked. It was much easier to test different cases and visualize what our program was doing using the simulator's graphical capabilities (see Figure 3). The interface includes a map of the world, a map of the active world around each instantiated robot, a dynamic map of the sonar values, and a window providing the current values of particular variables.

2. *Finding Bugs*

Some of the bugs that we discovered were found easily. The sorts of problems that were easy to find were the ones that were easily narrowed down to a particular line of code that was causing the problem. Common debugging techniques, such as printing out

the values of certain variables in a specified location within the code and using the debugger provided us with insight as to what was happening internally.

During simulation, we had some trouble with our code. The first part of the simulation process was to figure out how to get our program to connect to the robot simulator. Once that was done, we immediately ran into many difficulties. The first major problem we discovered was that, although we determined an appropriate direction to move in, we never issued a move command. Another problem that we found was that we switched between different units throughout the program constantly. This meant that many of our calculations were inconsistent in size, and therefore, meaningless. For instance, the robot didn't move in the direction that it was told to, because it expected degrees and was sent radians. Another motion problem that we encountered was that the robot drove in the direction relative to the robot and the target coordinates, instead of relative to the orientation of the robot. This problem caused the robot to continually rotate and drive in circles.

Once we fixed the movement problems, we discovered halting problems. The robot would never stop at the target, because it was going too fast and started orbiting the target point. After the halting problem was fixed, we discovered that some of the variables that we have set to constant values were set incorrectly. After we fixed them, we were plagued with "segmentation fault" errors. These errors were particularly grueling to pinpoint, because they would die in different places within the program each time. Eventually, we determined that it was a memory resource problem, and we tried to restructure out code so that it would be less memory intensive.

3. *Removing Bugs*

Some of the problems were relatively easy to remove. For instance, changing the units for a variable or a value for a constant were simple alterations. Another easy solution to a programming bug that came up included changing the angle of motion, within the call to the robot move function. It was a small change to make that angle be with respect to the robot's orientation instead of with respect to positions.

On the other hand, there were some errors that were quite challenging to fix. The worst of these kinds of bugs were the calls to robot functions that failed to work exactly as described in their manuals. The difficulty in fixing these problems arose because we had no understanding of what those function calls were actually doing. The manuals offered no further description and neither did the company web pages. We were just trying to get our program do what those functions said. We had no alternative robot functions to use. Almost by trial and error, we discovered a method to accomplish our initial goal.

4. *Remaining Problems*

The simulation is currently indicating some problems with our VFH program, but evaluation indicates that the problems which remain are inherent to the algorithm. There are some cases in which the robot crashes into the corners of obstacles or even tunnels through the obstacles when we run our program on the simulator. Also, there are tests where we put obstacles all around the robot, but at a large distance away. We then told the robot to move a short distance away to the target. Unfortunately, it would be unable

to move, thinking that it had no sectors clear. These problems arose because the algorithm immediately removes all explicit distance information. The only measure by which the algorithm tells the robot if it is safe to steer in a particular direction is by the sector certainty value.

The VFH algorithm is supposed to accommodate for distance to obstacles, because it increments the sector value linearly with the distance to the object. Yet, this certainty value also is incremented during each iteration in which an obstacle falls within that sector. Therefore, if the robot is driving slowly, then it will make more measurements, and the certainty value will increase substantially, just as if the obstacle were close. If the robot moving quickly, then fewer iterations will be made per time interval, and the certainty values will be smaller. In this case, the robot will think any obstacle within that sector is far away. If the robot is moving quickly, it may be too late to steer away by the time the robot recognizes an obstacle. This is why the robot sometimes collides head on with objects.

One way in which the VFH algorithm accommodates for sonar glitches is by using a smoothing function. When converting from the Cartesian certainty grid to the polar histogram, the certainty values are squared before they are mapped to the polar histogram. If the sonar interprets a garbage reading as an obstacle, then VFH smoothes the values within the sectors. This approach works well when the sonar reading was an inadvertent blip, but it also means that corners of objects are literally smoothed away. It becomes nearly impossible to represent corners of objects with this algorithm. This is why the simulation shows us that the robot has a hard time detecting corners; they were literally smoothed away.

The last sort of problem that we noticed was that there were times when the robot had plenty of room to maneuver around obstacles, but the robot failed to do so. The reason it behaved in such a manner is because it was reading values from all of the sonar, since the robot's environment had many large walls all over the place. Since the obstacles were distant from the robot, each certainty value was multiplied by a lower distance factor. Still, no matter how small the individual certainty values were, the sum of all of their squares ended up being very large. This value was often over the steering threshold, so the robot did not think the sector was a safe one to maneuver in. Once again, this problem is inherent to the algorithm, so there are many cases where distant large obstacles will prevent the robot from moving at all.

Chapter VI: Marcus

The last step of my project was to run the code on the robot, Marcus, and to see how well the implementation imports to the robot. Although our simulation worked for many cases, we were uneasy about the simulated cases that crashed the robot. We still wanted to import the code to the robot, though. We resolved to stay close enough to the robot to hit the big red button that turns the robot off, in cases when the robot did not work the way we hoped.

During this final phase of our project, we were not attempting to do any serious debugging, as we had done with the simulator. Instead, we were just trying to see if the tests we had run on the simulator worked the same as they had on the simulator. The times when there were obstacles that the robot did not detect were most likely due to the capabilities of the sonar. The sonar is fired at a fixed height and has a fixed height degree range that it reads. This range fails to see obstacles that are close if they are higher than this range.

Running the code on Marcus was quite gratifying. It was rewarding to see our code communicate directly with the robot. Best of all, Marcus successfully steered to the target locations in many cases.

Chapter VII: Evaluation and Conclusions

The process features that I evaluated include translating the VFH definition, using Marcus' facilities, and approaching navigation the way VFH does. This chapter discusses these few things. Finally, this chapter goes on to summarize my conclusions and recommendations for further research.

1. *Interpreting the VFH Definition*

There were times when using the VFH algorithm description was difficult. The VFH description mentioned the overall parts of the algorithm, but in some places the description failed to detail other important aspects. The algorithm uses many assumptions that are never defined or discussed. For example, it relies upon constant scanning to create a probabilistic distribution of obstacle representation. The algorithm fails to discuss the consequences of such world representations; however, there is no way that this sort of probabilistic representation can start with an accurate interpretation of the robot's world. Also, there are places in the algorithm's description where an idea is mentioned, but it is done so ambiguously, leaving room for dramatic variation in interpretation. This ambiguity and exclusion of information made it difficult for us to define our VFH specifications.

2. *Using Marcus' facilities*

The simulators themselves prove to be quite helpful. They provide a graphical interpretation of our program runs. We can watch the robot drive towards its target and

avoid obstacles that we have set up. When our program was not working properly, the simulator demonstrated what was really happening, and that simplified the process substantially. The simulator also has the benefits that it can not cause any physical damage to equipment or people when an erroneous program is run on it. Also, the simulator was helpful because it was so fast. It showed us results instantly; we could easily make a few changes to our program and watch the changes in action, using the simulator.

Although the simulator worked wonderfully, there were other components of Marcus' system that caused problems. The components I am referring to are the manuals for Marcus. These manuals state that the simulator and the robot will both understand and use the same robot commands. Then they go on to state that there are certain functions you must call to update the available robot information. For instance, there is an array that stores the sonar values, but the manuals state that the values in those arrays must be manually captured to update them. When we tested our program out, we ran into an enormous amount of confusion until we noticed that those arrays were, in some cases, actually being updated on their own. The consequences to the discrepancies between the manuals and the actual robot system had far reaching effects on our implementation of the algorithm.

3. Conclusions

The Vector Field Histogram algorithm has many benefits for navigation. It succeeds in providing fast collision avoidance measures. The fact that the program does

not stop in front of obstacles in order to maneuver around them is a major benefit over other navigational algorithms.

Although there is a significant processing speed improvement with VFH over other algorithms, there are drawbacks too. The VFH algorithm was described by Borenstein (13) with many constants. For instance, there were several constants mentioned in the paper that were used in calculations for velocity control and environmental representation. These constants needed to be tweaked, according to the paper, but the description never said in what ways. In fact, the paper defining VFH never gives a clear description of a range of validity for these constants, and it only states that these numbers need to be determined empirically. The VFH algorithm needs to have more straightforward and rigorous engineering documentation outlining its tasks. It needs to be defined more stringently, in order to understand all the intentions of the approach.

Of the straightforward and documented tasks involved in the VFH algorithm, there are some fundamental flaws. First of all, the algorithm has problems associated with how it represents the world. One of the first steps of the algorithm is to throw out all distance information, and from then on the algorithm tries to represent distance implicitly. Unfortunately, there are many instances when this implicit representation of distance is inappropriate and faulty. Some examples of these problems were discussed in the chapter on the simulation. Instances where an implicit representation of distance is inappropriate or faulty include environments with corners and large obstacles. Also, slow robot speeds can sometimes cause problems too.

4. *Recommendations and Continuing Efforts*

The overall intention of this algorithm was to accomplish fast collision-avoidance navigation, and VFH tried to accomplish this by reducing its data twice. This reduction of data dismissed all explicit distance information, and this made the information that the robot has to process per run much smaller. Indeed, there is so little to process per run that the robot does not need to ever stop to think about which way to move. Although this is a noble goal, it was not fully accomplished with this algorithm.

My recommendation is to reevaluate the assumption that dismissing explicit distance information is allowable. I found this algorithm worked rather miserably in far too many cases because VFH fails to recognize where items were within its world. I suggest that continuing efforts in this field of research should try to incorporate the goals of this algorithm with the techniques of navigating using explicit distances to objects.

Bibliography

History, Future, and General Information of Robotics

1. Dery, Mark “Robots Rising.” Documentary Film: Discovery Channel, March 8, 1998.

This documentary was wonderful for background information about robots! This movie presents information concerning the latest robotics projects in industry and research. I used it to get a better understanding of what kinds of projects are current in the field of robotics and for general inspiration.

2. “Hal’s Legacy: 2001’s Computer as Dream and Reality.” Ed. David G. Stork. The MIT Press: Cambridge, Massachusetts, 1997.

This book is a compilation of works from the experts in certain areas of computer science and robotics which discusses the visions of Clark and Kubrick’s 2001, and the hurdles that robotics will have to make it over before an automated HAL-like robot can be created. I used this book to understand the greater context of the field of robotics and the context for which a autonomous and mobile robot would fit.

Natural Vision and How the Brain Processes Visual Input

3. Minsky, Marvin. The Society of Mind New York: Simon & Schuster Inc., 1986.

This book discusses the role the brain has in thinking, and I used it to get a Minsky’s sense and description of what role vision has in thinking.

4. “Neurocomputing.” Ed. Anderson and Rosenfeld. The MIT Press: Cambridge, Massachusetts, 1989.

This is a collection of articles pertaining to neurocomputing and perception. The articles I found interesting and useful pertained to animal memory and models of the brain

5. Reisberg, Daniel. Cognition: Exploring the Science of the Mind New York: W. W. Norton & Company, 1997.

This textbook deals with cognition, which was helpful in understanding how the human brain processes visual knowledge and categorizes enough to make navigational decisions.

6. Hofstadter and Dennett, Daniel C. The Mind's I. New York: Bantam Books, 1981.

This book reflects on self and soul, and it was helpful in understanding another interpretation of consciousness and internal representations of surroundings.

7. The Mind's Eye. Readings from Scientific American, W. H. Freeman & Company, New York: 1986.

This collection of works focuses on the visual systems for several types of living creatures, including the brain mechanisms of vision. It was useful for understanding ways in which nature accommodates life-forms with an effective and efficient system to guide the creature throughout its world.

8. Crick, Francis. The Astonishing Hypothesis. New York: Simon & Schuster Inc., 1994.

This book very heavily focuses on the link between vision and intelligence. It aided me in learning about the visual systems for primates and basic principles we naturally use when interpreting visual input.

Navigation Techniques

9. Borenstein, Everett, and Liqiang Feng. Navigating Mobile Robots. Wellesley, Massachusetts: A K Peters, 1996.

This book was helpful in outlining the different visual systems for robots, in particular for use with navigation. It showed the benefits and pitfalls to these different methods of artificially representing the world around a robot, so the can see and avoid obstacles.

Computer Vision Research Information

10. UVA Vision Group “UVA Vision Group Computer Vision Research at the University of Virginia” UVA Vision Group <http://www.cs.virginia.edu/~vision/> (1996-1997).

This web page has links to the various components of the Computer Vision Research being done at UVA. I used it to understand the departmental context for my project.

11. Fisher, Robert B. "CVonline: The Evolving, Distributed, Non-Proprietary, On-line Compendium of Computer Vision" CVonline: The Evolving, Distributed, Non-Proprietary, On-line Compendium of Computer Vision http://www.dai.ed.ac.uk/staff/personal_pages/rbf/CVonline/CVentry.htm (27, October 1997).

This web page is a great place to start to research the various sub-fields of computer vision. It has links to every research field within and pertaining to computer vision. This source even has links to possible application descriptions. I used this source to get an overall sense of robot vision, as well as specific details in particular aspects of robot vision.

12. Huber, Daniel "Welcome to the Computer Vision Home Page" Computer Vision Home Page <http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/vision.html> (22 October, 1997).

This web page is a great source of information pertaining to robot vision projects and history. It has links to many topics pertaining to the field. A few of the links are to research groups, to general information archives, and to online publications. I will use this web page as a starting point for gathering lots of general information about robotics.

VFH Algorithm Details

13. Borenstein, Johann "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots." IEEE Transactions on Robotics and Automation. Vol 7, No 3 (June 1991): 278-88.

This document will be the one that I used most often during my thesis project. This paper defines the Vector Field Histogram Algorithm tasks, and I directly translated it to create the specifications for my project.

This document is the most relevant source for my project. It describes other navigational algorithms and compares VFH to them. I used this to get an understanding of reasons to advocate VFH over other navigational algorithms.

14. Borenstein, Johann and Yoram Koren. "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots." The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots. <http://www-personal.engin.umich.edu/~johannb/paper16.htm> (2 July, 1996).

This web page discusses the algorithm details in full. I used his page to get a better understanding of the algorithm which I implemented and evaluated.

15. Borenstein, Johann and Yoram Koren. "The Virtual Force Field (VFF) and the Vector Field Histogram (VFH) Methods – Fast Obstacles Avoidance for Mobile Robots." VFF and VFH -- Fast Obstacle Avoidance for Mobile Robots. <http://www-personal.engin.umich.edu/~johannb/vff&vfh.htm> (5 July, 1996).

This web page discusses histogram grids in detail, including several diagrams of these grids. I used this web page to get a better understanding of the histogram grids.

Robot References

16. Nomadic Technologies, Inc. Nomad 200 Hardware Manual. Mountain View, CA: Nomad Technologies, 1997.

This reference provides details about the robot's hardware. By using it, I was able to understand the physical layout and components of the mobile robot, Marcus.

17. Nomadic Technologies, Inc. Language Reference Manual. Mountain View, CA: Nomad Technologies, 1997.

This reference manual provides language rules appropriate for manipulation of the robot. It explains the important semantic and syntax components to functions that can be used to manipulate the robot.

18. Nomadic Technologies, Inc. User's Manual. Mountain View, CA: Nomad Technologies, 1997.

This user's guide describes what kinds of things I can do with the robot. I used it to learn about the available ways to manipulate the robot.

19. Nomadic Technologies, Inc. Nomadic Technologies, Inc. Home Page "Nomadic Technologies, Inc. - Merging Mind and Motion" <http://www.nomadicttechnologies.com> January 29, 1998.

I used this web page to try to find more information out about certain robot commands and to find out more about the company. It was dismally unsuccessful in clarifying any of the robot commands, but the robot pictures are neat.

Appendix A: Figures

Figure 1: Mapping active window cells onto the polar histogram

(figure straight from 15)

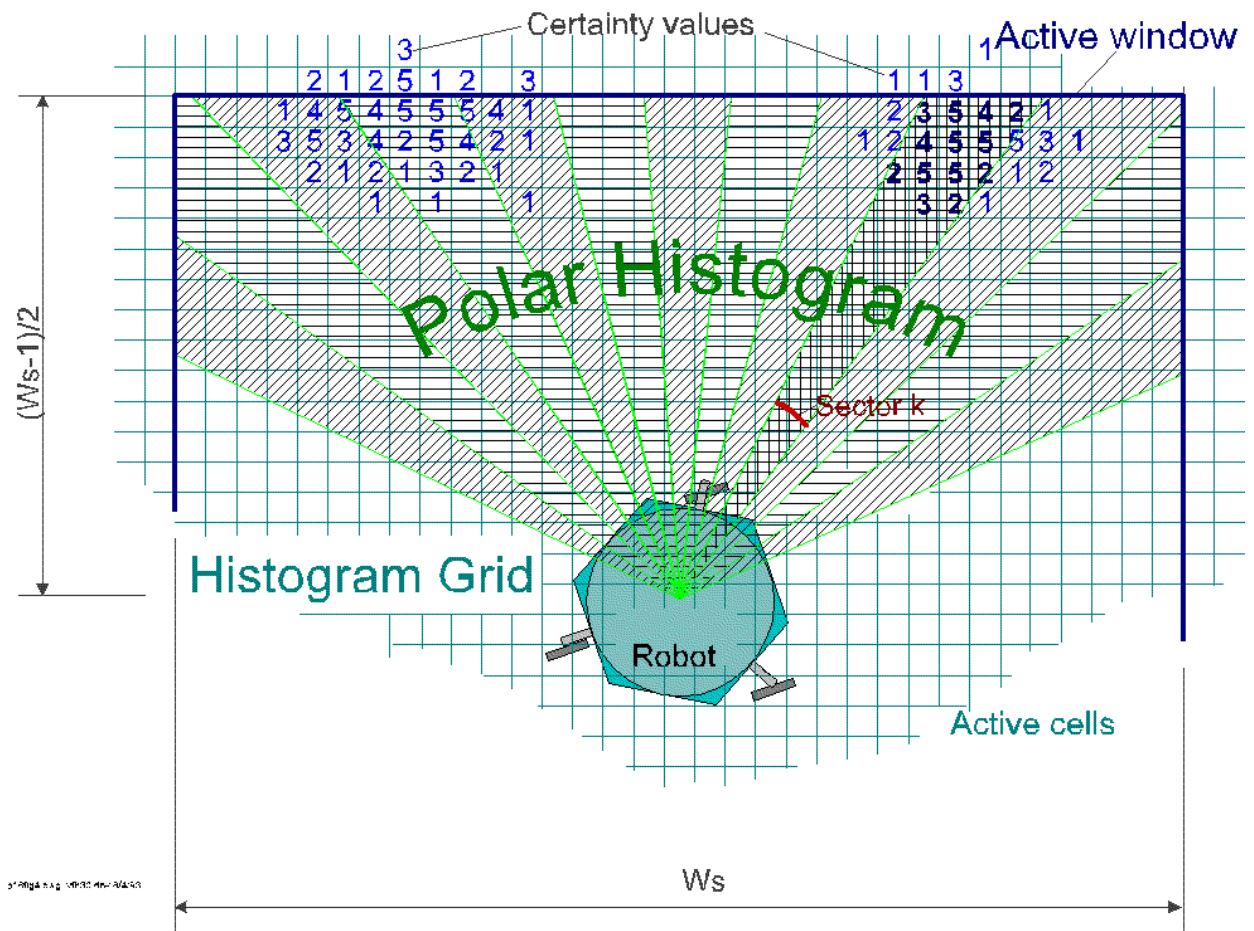


Figure 2: Selecting a safe direction of motion based on obstacle density

(figure straight from 15)

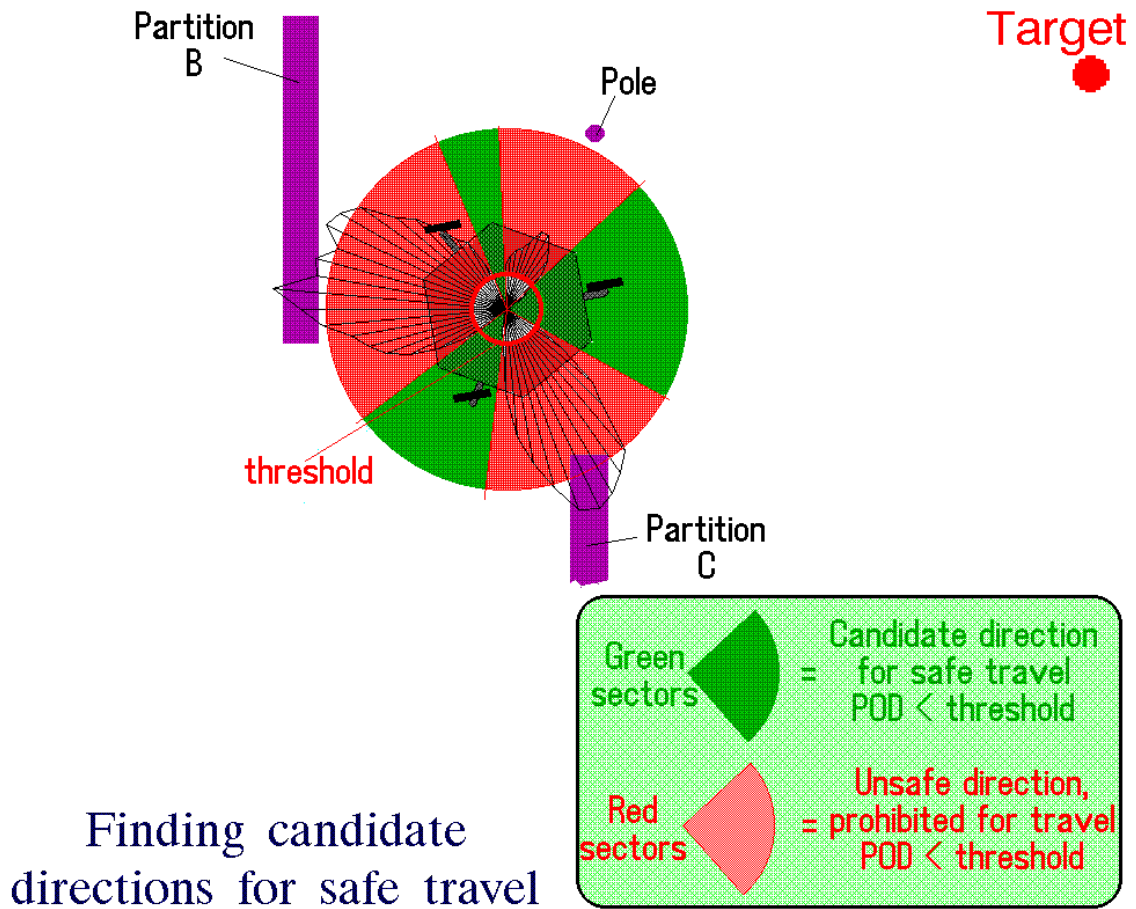
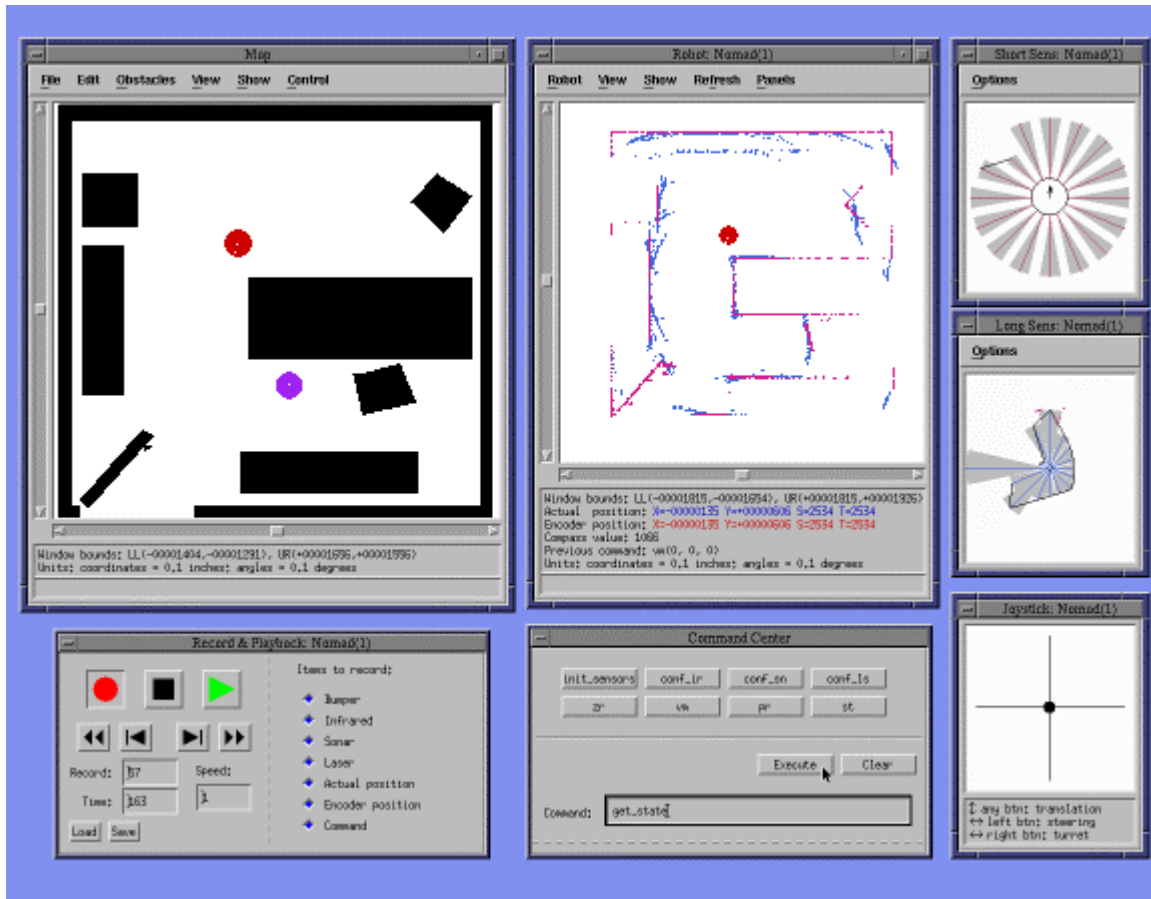


Figure 3: Screen-shot of the Simulator

(figure straight from 19)



Appendix B: Marcus, the Mobile Robot

Figure 4: Picture of Marcus

(figure straight from 19)



Statistics for Marcus

These statistics are taken directly from the Hardware Manual (16, 2-10).

Base

3 servo, 3 wheel synchronous drive non-holonomic system with zero gyro-radius*
radius = 18 inches (21 with bumper)
height = 30-35 inches

Motors

1 for wheel translation (max. speed = 24 inch/sec)
1 for wheel rotation (max. speed = 60 degrees/sec)
1 for angular position of turret

Processing System

shared memory multiprocessor system

master is a 486 pc

sensor interface is controlled by 1 or more Motorola MC68HC11F1 16 MHz controllers

motor control performed by Motorola 68008/ASIC three axis control system

processors communicate over master bus, an ISA bus

Power

840 watt hour removable package

Sonar System

16 sensors, equidistant around the circumference of the robot

each sensor can provide range information for 17-255 inches with 1% accuracy

Tactile System

20 independent pressure sensitive sensor switches on 2 bumper rings

each bumper ring has 10 switches that are interwoven with the switches on the other ring

360 degree coverage with 18 degree resolution

8 ounce sensitivity

* zero gyro-radius means that the robot can rotate around its center