

Algorithm Development for Behavior Based Research in Autonomous Robotics

Andrew Frick
GLCA/ACM, DOE ERULF
Albion College
Oak Ridge National Laboratory
Oak Ridge, Tennessee,

Prepared in partial fulfillment of the requirements of the Office of Science, DOE ERULF, and
GLCA/ACM program under the direction of Lynne Parker in Computer Science and Mathematics at
Oak Ridge National Laboratory.

Participant:
Signature

Research Advisor:
Signature

Table Of Contents

Abstract	3
Introduction.....	4
Material and Methods	
i) Equipment and procedures	5
ii) Algorithms	6
Result	8
Discussion / Conclusions	9
Acknowledgments.....	10
Figures.....	11

Algorithm Development for Behavior Based Research in Autonomous Robotics

Andrew Frick, Albion College, Albion, MI, USA 49224
Lynne E. Parker, Oak Ridge National Laboratory, Oak Ridge, TN 37831

Abstract

The Nomad200 robot is a research platform created by Nomadic Technologies Inc. The robot is a robust platform for many robotic tasks such as multi-robot cooperation and single robot research. The behavior sets created in this project are the basis for research in the cooperative robotic task of surveillance. The behaviors in the set are orthogonal in order to allow the robots to perform other self-contained tasks when inhibited from doing certain behaviors. These behaviors must also have the ability to handle real-world conditions and faults. An unreliable behavior is of little use for research outside of a simulator. The behaviors are a movement algorithm, a door finding algorithm, and an algorithm that detects objects moving through a doorway. These algorithms have mechanisms that allow the robot to distinguish between an actual doorway or object and a group of false data points. The fault toleration mechanisms are robust and simple allowing reproduction and use in multiple behaviors in the set. For the movement algorithm, direction vectors and vector algebra are used in a repetitive control loop to make up for inaccuracies in directional movement. For the other behaviors, a queue is used to maintain an average of the previous sensor readings. By adjusting the length of the queue the robot will become more or less sensitive to faulty sensors. Allowing the robot to deal with faults in the data will lead to a more stable control system that will be added in the future.

Research Category (Please circle)

ERULF: Physics Chemistry Engineering Computer Science Other_____

CCI: Biotechnology Environmental Science Computing

TYPE ALL INFORMATION CORRECTLY AND COMPLETELY

School Author Attends: Albion College
DOE National Laboratory Attended: Oak Ridge National Laboratory
Mentor's Name: Lynne E. Parker
Phone: (865)241-4959
e-mail Address: parkerle@ornl.gov

Presenters Name: Andrew Frick
Mailing Address: 1402 Charlton
City/State/ZIP: Ann Arbor, MI 48103
Phone: (734)665-4728
e-mail address: africk@albion.edu

Is this being submitted for publication?: Yes No
DOE program: ERULF CCI PST

I. Introduction

Computer Science Research in robotics is very different from research done with traditional computers. Creating algorithms on traditional computers works well because the hardware is usually reliable. It is often assumed that requests for data will always be processed and that information on the state of the system is accurate. This allows the algorithms to work independently of the hardware and analysis of the validity of an algorithm is not based on the reliability of the hardware. However, in robotics research, algorithms are judged as much on the ability to handle real-world situations as on theoretical soundness.

Algorithms must be proven in real-world tests and achieve the initial goals of the project to be considered sound. There is also much research that requires a black box theory, meaning that the algorithm will function appropriately in a given situation but will fail in many other known situations. It is the challenge of robotics research to remove this black box from algorithm development. Algorithms should be designed to handle any situation that is presented to them in a dynamically changing world. This requires a robust approach that will allow the robot to choose the appropriate action in a given situation. This problem does not have a trivial solution. Taking readings and matching readings to a set of predefined actions is not enough. There needs to be a large amount of error correction written into the software controls. Furthermore, controls need to be more general, allowing them to work in a large number of situations.

The high-level goal of this project is to achieve a fault-tolerant multi-robot system that can perform the task of group surveillance. This would be useful to develop for situations where a building or exit needs to be monitored but the conditions are too dangerous or too hazardous for humans to perform. Systems of this nature rely on a group of predefined behaviors that are

chosen by the robot, depending on the situation. Behavior, in this sense, is being used to describe actions taken by the robot in an autonomous nature. Once the human operator gives a high-level command, any movement done by the robots' software will be considered a behavior. By developing the behaviors separate from the group controls, the task can be changed just by changing the natures of the behaviors. Since the control structure is getting information from and performing actions through the behaviors it is important to create behaviors that are robust and fault tolerant allowing the end result to be achieved autonomously.

This project developed two behavior sets and the algorithm for a third. The main difficulty with each problem was dealing with inaccurate sensors. The three behaviors addressed in this project are a doorway finding, a doorway monitoring, and point-to-point movement algorithms. The point-to-point movement behavior deals with the inaccuracy of the dead reckoning ability of the robot, since the farther it moves the farther it will be from its expected location. This forces the programmer to install fail-safes in the software so that the robot rechecks its position and direction before it moves very far. The door finding behavior uses a wall following algorithm and modifies it to recognize where the doors are. This approach is difficult since an inaccurate sonar reading reads identically to a no return sonar reading. The sentry algorithm is based on the error correction for the door finding algorithm with a few changes allowing it to work in that capacity.

II. Material and Methods

(i) Equipment and procedures

The equipment used in this experiment consisted of several components. A Sun Sparc 20 workstation running the Sun Solaris operating system was the main computer used to write and test the software before being tested on the robots. The manufacturer of the robots, Nomadic

Technologies Inc. provided the simulator software that was used on the Sun workstations. The robots used are Nomad200 scientific research robots. The robots are cylindrical in shape and about three feet tall (figure 1). The drive system on the robot consists of three wheels (one for direction and the others for propulsion). Each robot is equipped with two sonar arrays each consisting of 16 sonar. The upper sonar (figure 2) is the main sensor used in this project. These sonar are audio sonar and can accurately detect objects that are within about five feet (about 2 meters) of the robot. The other array consists of infrared sensors that are lower down on the robot and are used mainly when working in a very confined space. The robots have onboard computers that can be accessed through the Sun workstation through a Cisco Systems radio Ethernet. Two of the robots have Pentium class processors and two have 486 class processors.

The development of behavior dealt with developing the algorithm or a starting point for the algorithm. Algorithms are then implemented using the C programming language and a library of functions that were provided by the robot manufacturer. Programs are compiled on one of the Sun workstations using the simulator. This allows the programmer to find and debug problems with his code without the risk or time involved with using a real robot. Once a behavior is working flawlessly in the simulator, the code is recompiled for the robots. Tests are then performed with the actual robots where any real-world specific problems can be addressed. The real robots also face the challenges of inaccurate sensors and dynamic changes to their environment.

(ii) Algorithms

All movement algorithms in this section had an obstacle avoidance algorithm embedded within them. If the robot was moving and an object blocked its path the robot would attempt going around and would then resume its previous task.

The point-to-point algorithm accounts for the errors that occur during long movements by the robot. The algorithm that was developed uses some vector algebra and a small control loop to constantly check the orientation of the robot and its position to the target. This is done through the use of a history variable. The robot has data on its current position, its previous position and its destination. By creating vectors from the previous position to both the current position and the end location, the robot can determine if it is pointed at the target. By using the cross product of these two vectors, the robot determines how far off course it is and what direction to turn. This procedure is looped until the robot gets a certain distance from the destination and then it will stop.

The doorway finding algorithm is a little more complex. This algorithm uses a group of 4 sonar that are placed along the side of the robot. The wall following part reads the sonar and compares them to one another. If the first sonar has much higher reading than the back sonar the robot realizes that it is drifting away from the wall and will adjust accordingly. If it approaches a corner the sonar on the front will alert the robot of a wall and it will slow or stop and turn until there is no longer a wall in front of it.

The challenge is determining the difference between a doorway and a corner. The algorithm monitors most forward side sonar to see when it stops registering a wall. When this occurs the robot will then move forward at a slowed pace and look for the next sonar back to recognize the opening. When this occurs, the robot then decides there is a door and waits until the third and fourth sonar have picked up the wall again to mark the end location of the door. To help deal with errors in the sonar, the program keeps a history of the last 30 readings. The program also waits for the average on given sonar to change rather than a few readings. This prevents one or two false data points from putting the robot into the subroutine that actually finds

the door. The average is created using a queue so that when a new data point is taken all of the data does not need to be recalculated. By subtracting the amount of the data point that is being replaced from the previous total and then adding the current data point, the total can be kept accurately without having to retake the sum of the data points every time new data is collected. Since the program is monitoring an average, the number of data points taken into account in this average can be easily changed to adjust the sensitivity of the robot to inaccurate sensor readings. A balance needs to be reached because if the program is too sensitive the robot produces inaccurate results. However, if it is too insensitive, the robot will not recognize smaller doorways and be ineffective.

The door-monitoring algorithm can be used by the robot to stand sentry next to an open door. It uses the average system that was implemented in the door finding system. The difference is that for this algorithm all of the sonar will be monitored. For this algorithm the current sensor data is compared to the average rather than just using the average. Most inaccurate sonar readings register as a long reading, so inaccurate readings would not give a false positive the way they do with the door following program. The average allows the robot to record the condition of its surroundings so if one of the sonar has a shorter reading than the average for that sonar then the program will alert the human controller.

III. Results

The results of the project are mixed. The point-to-point behavior worked in the simulator and then translated to the real world very well. The robot was able to move to any point that it was given and the obstacle avoidance part of the algorithm also worked well. Figure 3 shows the robot executing the point-to-point behavior in a real world environment.

The behavior was implemented in the simulator and was tested on rooms with varying sizes and numbers of doors. The program worked flawlessly in the simulator. Figure 4 shows the trace of the robot on one trip around an example room. Figure 5 shows the output of the program, which marks the beginning and end of each of the 3 doors in the room. However, when the program was compiled and run on the real robots, the behavior never worked properly. The tests were done to set the robot up and allow it to try and find a door in a wall. The robot found the door a few times but was unable to repeat any successes. Since there were many problems with the implementation, most of them could be traced to how the sonar was dealt with. After each run a number of adjustments were made and the test was run again. In most instances, the failures were the robot missing the door entirely or turning into the open doorway and then exiting the room.

The sentry behavior was not implemented and will remain as a future research task.

IV. Discussion / Conclusions

This project is not complete. The work done on the behavior sets is far from complete. The point-to-point behavior was very successful and the algorithm was robust since it worked on the real robot on the first try. The door-finding algorithm was proved successful in the simulator. The wall-following part of the behavior works very well. Problems arose in the adjustments of the queue that helped with error correction. When the queue was too long, the robot would drive by the door and when it was too short, the robot would simply turn into the doorway thinking it was a corner. Another situation that caused trouble was when a sonar gave false readings right before reaching the doorway. In this situation, the queue worked against the robot since there was not a noticeable difference in the values contained in the queue. The few successes were due to a lack of faulty sonar readings on those runs.

The problems with the sonar can be fixed with more time and more adjustments to the sensitivity of the robots sensor. A balance was never found between a queue length that was too long and one that was too short. Speed could also have been a factor. By changing the speed of the robot, the level of error could go down. The material that the sonar was reflecting off of could have contributed to the failure of this algorithm. The fabric cubicle walls that were used to make the rooms for the robots do not reflect sonar as accurately as other materials.

These behaviors are the basis for more advanced research in distributed robotics but without fault tolerant behaviors the higher-level work becomes more difficult. Even though the multi-robot system was never operational, this project provides a strong start to research to that end. Once the bugs in the door finding behavior are ironed out, the behavior will be stable and robust enough to be used in a multi-robot system.

V. Acknowledgements

This research is sponsored by the Engineering Research Program of the Office of Basic Energy Sciences, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

I would like to thank the United States Department of Energy for providing the funding and resources at the Oak Ridge National Laboratory. Without this the research could never have been conducted.

My thanks also go to my mentor Dr. Lynne E. Parker for her guidance and teaching throughout the project. Special thanks go to Sarun Teeravechyan and Tina Lanning for their help with code problems and writing, respectively.

Figures



Figure 1: The Nomad200 research robot.

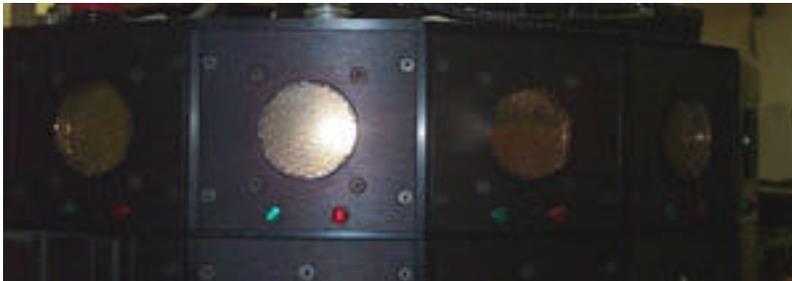


Figure 2: The sonar that was used for in all of the algorithms in this project.



Figure 3: The robot performing the point-to-point behavior

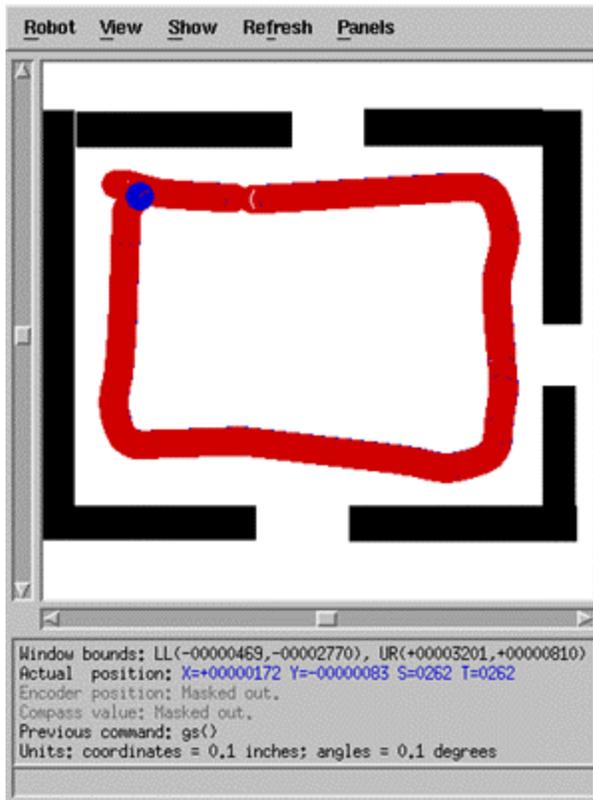


Figure 4: The trace from the simulator. This shows the robot completing a trip around the room and finding the doors.

```
The server is running
Begin door search
26 0 1
Beginning of door found 1128 -95
End of Door found count 1
X: 1654 Y: -63
Begin door search
23 0 1
Beginning of door found 2564 -893
End of Door found count 2
X: 2600 Y: -1341
Begin door search
27 0 1
Beginning of door found 1708 -1819
End of Door found count 3
X: 1054 Y: -1742
```

Figure 5: This is the command line output of the robot behavior when it was run in the simulator. Notice that it found all three doors and marked their locations.