Robot Communication

6/25/07 to 8/09/07
By Benjamin Shih
Montgomery Blair High School
bs1212@gmail.com

Executive Summary

The goal for my summer project was for two robots to work together and communicate with each other to map a room, and find a “finish” point. I was also interested in improving the communication algorithm. However, due to partially available equipment and limited time in the summer, I was only able to accomplish part of this interesting summer project.

I have been trying to work with two MobileRobot AmigoBots in the University of Maryland (UMD) Hybrid Network Laboratory. I encountered many problems, including the wireless connection, incompatible camera, and the C++ compiler.

After fiddling around with the parts and trying to fix the problems, I concluded that I would not be able to operate both robots simultaneously. My focus was shifted to making one robot operate autonomously. I was able to collect all the functioning pieces from both robots and put them on a single robot to make one robot function. I also developed programs which allow the robot to perform some basic movements, including turning and wandering. In the end, I identified a list of parts that are needed for future research and documented a manual for operating the robot.

I. Introduction

Today we can find all kinds of robots applications in our daily life. These applications range from basic, such as civil simplifications, to advanced, such as biomedical engineering and space exploration. Conventionally, a single robot has been used to accomplish certain specific tasks; however, multiple robots have the potential to accomplish a task more efficiently and effectively. This is very important, especially when time is a critical component. Some robots require assistance from others, such as a land roving robot getting vision of the other side of a mountain. In addition, if two robots needed to scout an area, they could do so without overlapping paths. The major challenge is to develop artificial intelligence, so multiple robots can be grouped together and can communicate with each other and work together to accomplish assigned tasks on time.

In section II, I am going to discuss the approach I used to operate and program the robot. The task was difficult because I had originally planned to have two robots communicate. In section III, I am going to discuss the results I obtained. I was able to connect the robot to the Internet and program it. In section IV, I provide detail discussion on the problems I encountered; parts were missing, and software was faulty. Section V is about the implication of this summer project and the importance of robot communication in the future. Section VI discusses what I would like to do to further the study of robot communication.

II. Approach

The two robots, in the UMD Hybrid Network Lab, consist of a polycarbonate body and aluminum chassis. They are equipped with computers, both of which run Debian Linux. The computers include basic ports, such as monitor, keyboard, and mouse. A wireless modem is attached inside. They also include iSweet cameras plugged into USB ports. The computers share a battery with the robots that they are connected to. Each robot has eight sonar that allow it to detect obstacles.

I connected the computer on the robot to a monitor. The computer already had programming software installed, so I studied the code and wrote my own programs. Setting up the network allowed me to remotely control the robot. I concluded that I would not be able to operate both robots simultaneously, so I decided to make one robot operate autonomously. I was able to collect all the functioning pieces from both robots and put them on a single robot to make one robot function.

III. Results

I could not fulfill my original goal because one of the robots could not be used, and sufficient documents for robot operation did not exist. Instead, I focused on operating one robot and wrote a manual for operating this type of robot. This manual documented steps to connect the parts of the robot and functionalities that the robot had. Documentation of problems faced by previous users would help others to avoid mistakes and save time. Not knowing what materials were actually needed to program the robot made getting started difficult. The following list includes the major results.

- Simulation
- Recharging
- Running Pre-Set Programs
- Basic Programming
- Networking

Simulation

MobileSim allows the user to simulate a user-selected program by performing the operation on a digital map. For example, assume the digital map has a wall and the program run tells the robot to only move forward. The wall is also in the path of the robot. The robot will move forward until it hits the virtual wall, and it will continue to run into the wall until the program is terminated. The programs simulated what they are designed to do, using digital maps that have been created.

Recharging

After I first drained the batteries of the robots, the indicator started flashing red. I was not sure if that meant something was wrong with the connection or if the battery was low. I left the charger in overnight because the battery port was specifically designed for the robot. The next day, the indicator turned green again. Therefore, a red indicator means low battery, a flashing red indicator means extremely low battery, and a green flashing indicator means sufficient battery.

Running Pre-Set Programs

The Aria program had already been downloaded onto the robots. It included a demo program, which allowed the user to test its sonar, configurations, battery, camera, system. It also included other tests such as battery and sonar tests. The programs were written in C++, and they came with a compiler/debugger. Once a program was compiled, it could be run simply by typing its name in the command line.

Basic Programming

I had moderate success with programming. Moving forwards and backwards was simple because they only required a few of parameters. I had also played with the programs that allow the robot to travel a given distance, travel in a triangle, and travel in a rectangle.

The most important part to programming the robots was to remember all the other code that was mandatory – a lot of the codes in the beginning and end remained the same in all programs. Some of the programs include:

- Wander – Makes the robot move around, avoiding obstacles detected by its sonar.
- Turn – Makes the robot turn.
- Teleop – Allows the user to control the robot.
- Sonar – Turns on all eight sonar and returns the frequency they detect.
- Server – The computer can "talk" with the robot. A certain input will cause the robot to perform an action.
- Move forward/backward – Move the robot forwards and backwards. Some sample code is provided below.

```
…
ArActionConstantVelocity constantVelocity("CV", 400);
…
 robot.lock();

 robot.comInt(ArCommands::ENABLE, 1);
 robot.addAction(&constantVelocity, 25);
 robot.unlock();

 robot.waitForRunExit();
 Aria::exit(0);
}
```

Networking

With the help of George, I was able to connect the computer on the robot to the internet through an Ethernet cable. The main problem was that I had been using a faulty cable to connect to the modem. Once the cable was changed, the connection just had to be enabled.

Through additional help, I was also able to connect the computer on the robot to the internet through the UMD wireless network. I had to change the interfaces file so that the robot would recognize the wireless signal.

However, this wireless network posed another problem. The objective of the network was to work on the robot from another computer. The wired network in the lab was not secure, but the UMD network was. Because of this, the computers in the lab could not connect to the robot. In the end, I just had to connect to the UMD network with the computer I was SSHing with to be able to SSH to the robot.

IV. Discussion

Working on the AmigoBot project has been challenging and time-consuming because of the following problems I have encountered:

- Equipment Location
- Camera
- Battery
- Defects
- Java
- Contacts
- Software

Equipment Location

The first predicament was the location of all the equipment. Aside from the robots, the rest of the equipment was scattered throughout the room. Only one charger was found. I also had a hard time figuring out how to operate the machine because the company's manual is poorly written and misleading. It told me to download software to a computer, and use an AmigoLeash to connect to the robot through the serial port. Pedram Hovareshti helped me to find what we thought was the leash. I spent the first week following its instructions. I downloaded MobileEyes, MobileSim, and Aria, the software for operating the robot recommended by the company, onto a Windows computer. The Aria program consisted of many programs in python, Java, and C++ that would make the robot perform tasks. However, none of them worked. MobileEyes did not work either because it required a robot server, which no one in the lab knew. MobileSim was the only program that successfully operated. Using MobileSim, I was able to run programs and simulate the movements of the robot. However, it was not helpful because I wanted to physically navigate the robot throughout the room, not through a computer generated map. There were also no instructions about the computer attached to the top of the robot. At the beginning of the third week, Pedram took me to visit another lab, which was working on a more complex robot built by the same company. They told me that I should directly connect the top computer to a monitor and work from there.

Camera

The cameras did not match the options provided in the demo program. In the program, when I select the camera mode, it offers me the choices of Sony PTZ, Canon VCC4, DPPTU, AMPTU, and inverted Canon VCC4 cameras, either connected through USB or serial ports. However, the cameras attached to the two robots are clearly labeled iSweet. iSweet specifications stated that the camera was compatible with the following software: iChatAV, QuickTime Broadcaster, Yahoo Messenger, BTV, iStopmotion, and iVisit. None of these software were seen on the robot. These options implied that the robot was able to pivot the camera and look around. However, the iSweet cameras were firmly locked in one position.

After loading the GUI on the robot, I discovered that it had a program installed called ACTS (Activmedia Color Tracking System). ACTS allowed me to see using the camera. It claims to be integrated with Aria, but I could not use it in a program.

Battery

I had also had problems with the battery. Through tests, I learned that the entire unit could only be charged by connecting the computer and robot, but the robot stored the energy. In addition, the computer could only use the energy from the battery in the robot, and can not operate on external charges. The batteries were also drained rather quickly.

Defects

Both of the robots contained defective parts. The first robot, which will be referred to as Robot A, has a broken mouse port. It also had problems running programs. Whenever I attempted to run one, it displays the message: *Aria: Received signal 'SIGSEGU'. Shutting down.* I did not understand why this had happened, but one of the differences from the other computer was that when I logged into an account, it displayed *at76c503.c: using BSSID 02:00:a9:d7:1e:00.*

Robot B had a weakened battery. It retained less energy per charge, and the hard drives shut off by themselves when they did not have enough energy remaining. Robot B does not have the problem Robot A does, but because it has a bad battery the robot can turn off in the middle of programming.

To resolve this issue, I disconnected the wires connecting the robot to the computer, and swapped the connections. However, this also caused a problem for testing programs, because movement was severely restricted when the two robots are connected, because in order to move one would have to drag the other. Later, I unscrewed and switched the parts so that at least one robot operated successfully.

Java

I originally planned to program in Java. The installation of a java wrapper, which allows the compiler to read java as well, requires a java runtime environment and other programs. The process is complicated, and it is not worth using.

Although the Java invocations of methods are much simpler and easy to read, adjusting to the C++ code was not extremely difficult.

Software

I had trouble with a lot of the software. When I was working on the Windows computer, I could not operate the MobileEyes and Aria programs. I also had problems with the ACTS program for the camera.

After the network was repaired, I started having issues with compiling a C++ program on the robot. The problem seems to be that G++ is not installed. When I try to install it, the robot tells me that I need to update other software. When I try to update the software, it tells me I need to update Linux and gives me a list of packages that I can install. However, none of these packages work, so I can no longer compile programs on the robot.

Contacts

Contacting past students who had worked with these robots had also been a problem. I had been given contact information for Dion Blazakis and Sung Park. My email to Dion contained where I was working and what I was working on, but he only responded after Pedram sent him an email. Dion was the lab manager, so he told me that he did not know much about the work done on the robots. I left a message and emailed Sung, but he never responded.

In summary, I was not able to achieve my original goal due to the issues mentioned above. However, the implication of the robot communication could support a lot of dangerous activities that human faces today. I think this project should continue to be explored. I have identified next steps for the future research provided resources.

V. Implication

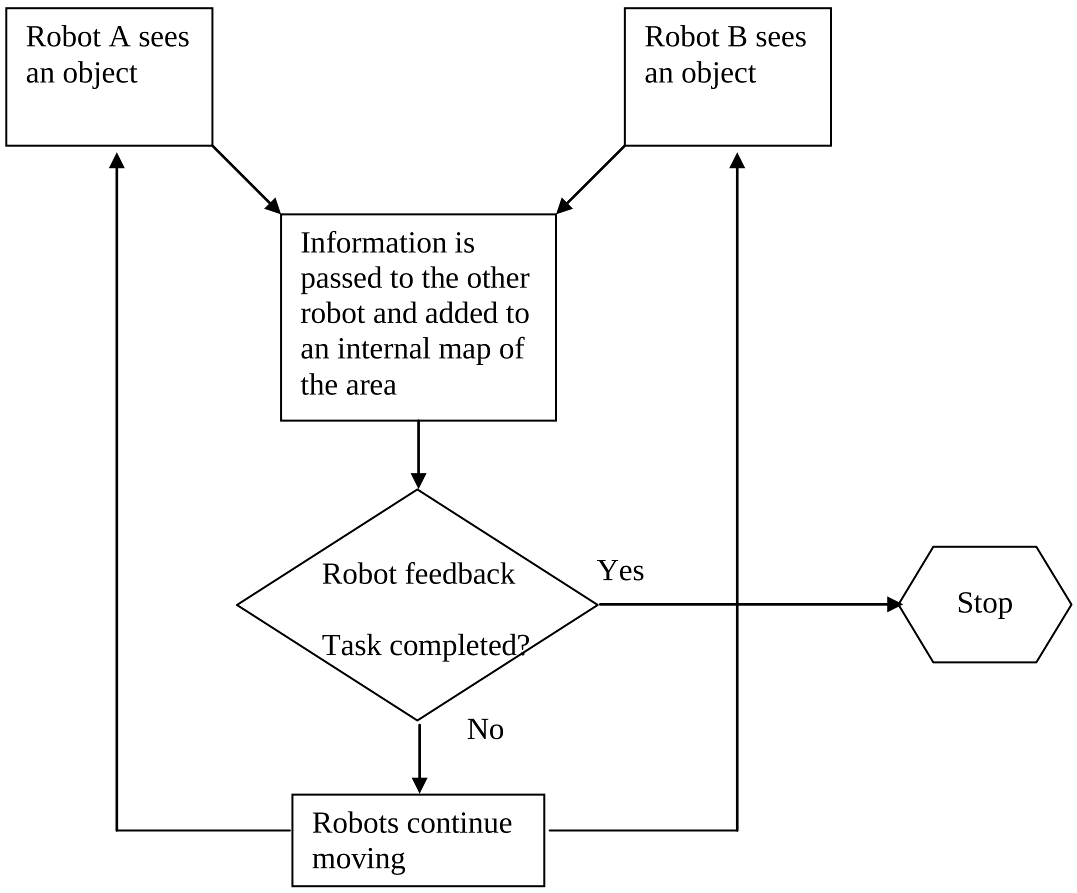
Robots working together can do many things. Their combined efforts can help navigate terrain. For example, if one robot found an obstacle, another robot could fly up to check for the best way to get around the obstacle. Communication could also be used in deploying robots in war. Instead of risking human lives, robots can be sent in to replace marines. Communication would also be useful in natural disasters. They can work together to find and rescue people.

VI. Next Steps

These commercial robots already have algorithms to gather information and work together. See Graph 1 for a simple flow chart for how I believe two robots will operate to map a room. However, no algorithm is perfect, and many things can be done. For example, the algorithm could be more efficient or simplified. The algorithm could be improved by:

- Stronger signals to prevent interruption
- More transmissions per second to ensure that no information is lost or repeated
- Incorporation of mass robots for operation

Graph1. Flow Chart of Cooperative Robot Algorithm



References

Robot:
http://robots.mobilerobots.com/amigobot/amigofree/AmigoGuide.pdf
http://robots.mobilerobots.com/amigobot/originalAmigos.html

VI editor:
http://www.eng.hawaii.edu/Tutor/vi.html

Debian Linux:
http://www.ss64.com/bash/
http://www.debian.org/doc/manuals/user/ch6.html

Networking:
http://www.debian-administration.org/articles/254
http://www.debianadmin.com/debian-networking-for-basic-and-advanced-users.html
http://qref.sourceforge.net/Debian/reference/ch-gateway.en.html

Camera:
http://robots.mobilerobots.com/ACTS/

Acknowledgements

I would like to thank Professor Baras, for valuable discussions and guidance in this interesting field. I would also like to thank him for providing me this excellent opportunity to work in the Hybrid Networks Lab. I have learned tons about Linux and robot algorithms during this extremely short summer. These experiences will be valuable to carry on into my high school years and further higher educations. I would also like to acknowledge Pedram Hovareshti, for consistently supervising and encouraging my work as well as Dr. Yadong Shang, for endlessly helping me with my work. Finally, I would like to thank my parents, for driving me to the University of Maryland everyday this past summer.