

Multi-Robot Navigation and Coordination

Daniel Casner and Ben Willard

Kurt Krebsbach, Advisor

Department of Computer Science,

Lawrence University, Appleton, Wisconsin 54912

daniel.t.casner@ieee.org, benjamin.h.willard@lawrence.edu



Abstract

Moving from a single to multiple robots increases the resources available to accomplish a task but it also adds its own complexity as the robots must be coordinated with each other to function efficiently. Exploring an environment is a task particularly well suited to multiple robots. This paper describes a test project involving navigation of two robots in a landmark rich environment to travel to a common location.

Introduction

Coordination between multiple robots to accomplish a task more efficiently is currently the subject of a great deal of

robotics research. With the opportunity to use two AIBO robots this term we wanted to experiment with multi-robot coordination in a simple project which could be conducted during a single term. Thus we choose to have the two robots attempt to navigate to a rendezvous location while communicating with each other.

Domain

For our experiment we choose the quasi-circular hallway of one of the academic buildings on campus. We placed unique landmarks at regularly spaced intervals in the environment to make it easier for the robots to localize themselves. Because the domain is a fairly narrow hallway and is effectively free of obstacles, it can be represented as a one dimensional space for purposes of navigation.

Robots

We used two Sony AIBO robots for this project, one ERS-7 and one ERS-220. The AIBO is well suited for small robotics projects like this. It is fairly sensor rich, can easily travel over the distances we are interested in, has built in 802.11b wireless communication and is programmable in C++. The specifications for the two models are slightly different and are quoted below [1].

For the ERS-7:

- 576MHz MIPS R7000
- 64 MB RAM
- 802.11b wireless Ethernet
- MemoryStick reader/writer
- 18 PID joints, each with force sensing
 - 4 legs
 - 3 joints each (elevate, rotate, knee)
 - 1 paw button each
 - 3 joints on neck (tilt, pan, nod)
 - 2 joints on tail (tilt, pan)
 - 1 joint on mouth
- 2 ears, 1 Boolean joint each (flick up or down)
- 26 independent LEDs
- Video camera
 - 56.9° wide and 45.2° high
 - Resolutions: 208x160, 104x80, 52x40
 - 30 frames per second
- Stereo microphones
- 3 IR distance sensors
- X, Y, and Z accelerometers
- 4 pressure sensitive buttons (one on head, three on back)
- 1 Boolean button under mouth
- Sensor updates every 32 ms, with 4 samples per update.

And for the ERS-220:

- 384 MHz MIPS processor

- 32 MB RAM
- 802.11b wireless Ethernet
- MemoryStick reader/writer
- 15 PID joints, each with force sensing
 - 4 legs
 - 3 joints each (elevate, rotate, knee)
 - 1 paw button
 - 3 joints on neck (tilt, pan, roll)
- 1 Boolean headlight
- 20 LEDs
- Video camera
 - Field of view 57.6° wide and 47.8° high
 - Resolutions: 352x288, 176x144, 88x72, 44x36 pixels.
 - Up to 25 frames per second
- Stereo microphones
- IR distance measure, aligned with center of camera
- X, Y, and Z accelerometers
- 2 Boolean buttons on head
- 1 Boolean button under chin
- 4 Boolean buttons on back
- Sensor updates are sent every 32ms, with 4 samples per update.

Tekkotsu framework

To speed up the development process we used the Tekkotsu development framework for AIBO robots [1] developed at Carnegie Mellon University. Tekkotsu provides midlevel functionality to the programmer. By using it we started programming with functions at the level of abstraction of walk forward at 10 centimeters per second and what pixel is the center of a color region at, instead of having to control each servo individually and develop the vision system from scratch.

Landmarks

Our landmarks consisted of sets of four squares of colors chosen to have high contrast when picked up by the robots' cameras. With four colors 24 unique combinations are possible so we manufactured 24 landmarks such that when distributed evenly along the length of the hallway there would be a landmark about every 3.5 meters. The landmarks were placed only along the inside wall of the circular hallway so that by noting which side of its body the robot detected a landmark on it would know if it were traveling clockwise or counter-clockwise around the hallway.

Implementation

Communication with another AIBO via WiFi

1. Network Implementation

At the lowest level, communication between the two robots was made possible by their support for 802.11b wireless networking. More specifically, we were able to make use of the *wireless* class supplied by Tekkotsu in order to send commands from one robot to the other. The *wireless* class has built in methods that facilitate the opening, closing, and maintenance of sockets over the wireless network. The *wireless* class supplies a socket object, and the socket object recognizes standard socket operations such as *printf* and *write*.

2. Commands and Arguments

When a transmission occurs between the robots, it is always in the form of a command followed by an argument (figure 1). In the cases that no argument was required, a standard NULL argument was sent with the command. The leader can issue a number of commands to the slave. The leader can instruct the slave to localize,

navigate to a specified landmark, give a status report (the current state of the slave), or stop. The slave can only issue status report commands to the leader. These commands include updates on location and status. For instance, the slave might inform the leader that it has reached LANDMARK 2 by sending the command LOCINFORM with an argument of 2. The leader accepts this command, updates its records regarding the slave's state and location, and issues a new command to the slave based on the state of both the leader and the slave.

| Command | Argument(s) |
|------------|---------------|
| STOP | NULL |
| GOTO | Landmark Code |
| LOCATE | NULL |
| LOCINFORM | Landmark Code |
| DESTINFORM | NULL |
| SHUTDOWN | NULL |

Figure 1: Commands and Arguments

Cooperation and State Machines

1. Robot Cooperation

In order to facilitate cooperation, we used a master slave configuration. The leader robot is responsible for creating a very basic plan based on both its state and the state of the slave. At each step of the plan, the leader communicates to the slave the actions that the slave must perform to move closer to the goal state. Both robots are dependant on each other to a large degree. The leader is dependant on status and location reports issued by the slave because the slave's state is a necessary consideration in the formulation the rendezvous plan. Likewise, the slave is dependant on the leader because it cannot "think" for itself. Instead, the slave relays information about its environment to the leader and relies on the commands issued by the leader in order to reach the goal of a rendezvous.

2. State Machines and State Maintenance

Each robot maintains a record of its current state (Figure 2) but the leader maintains a record of both its state and the state of the slave. The leader is able to make educated decisions because it examines the state of both robots first. Transitions through the state machine are driven by commands issued from the other robot and also by objects perceived in the environment. State diagrams in figures 3 and 4 illustrate the state machines for each robot.

| States |
|--------------|
| STOP |
| READY |
| LOCALIZING |
| LOCALIZED |
| MOVINGTODEST |
| ATDEST |

Figure 2: Possible States

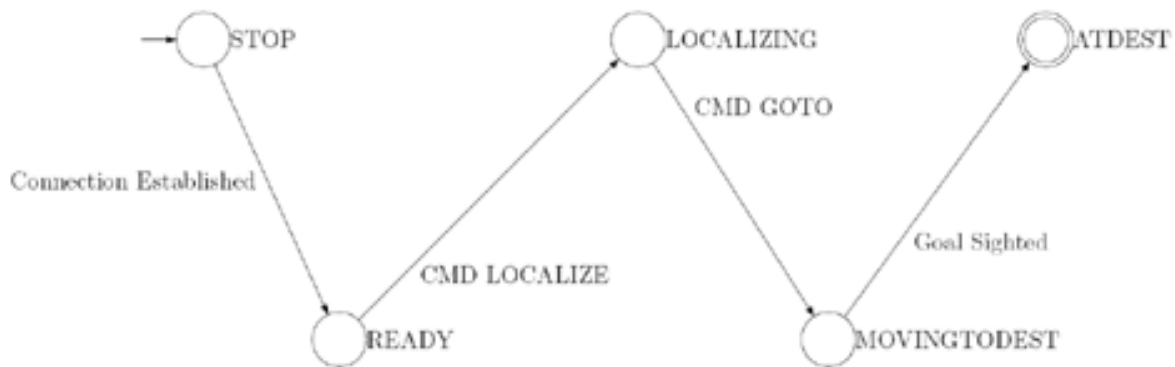


Figure 3: Basic State Diagram for Slave

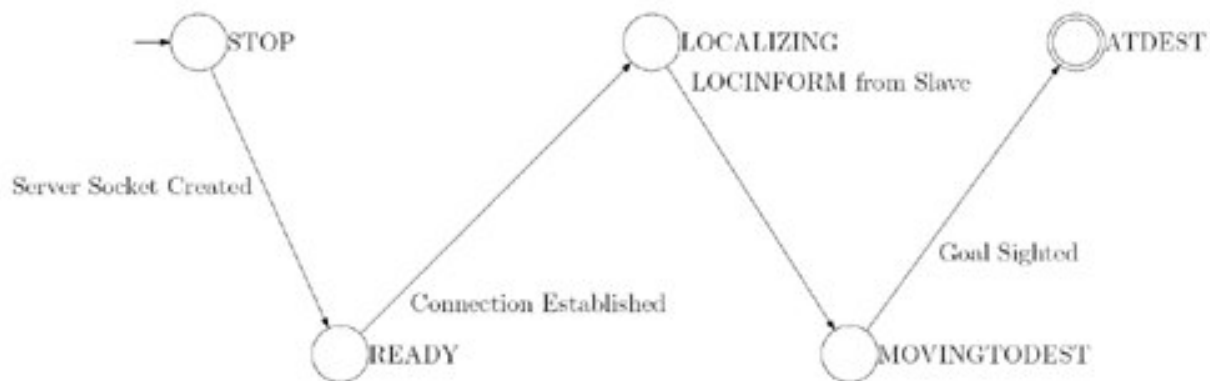


Figure 4: Basic State Diagram for Leader

Basic Hallway Navigation

Both robots use the same relatively simple algorithm to avoid crashing into walls. Data about wall and distances is collected through the long distance IR sensor. When the robot is walking down the hallway (in either the localizing or navigating state), it continually pans its head from the left to the right at a speed of 1.5 radians per second. IR measurements are recorded when the head is facing left, center, and right (Figure 5).

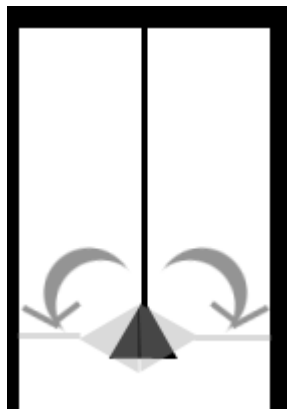


Figure 5: IR Measurements

These measurements are assumed to be current, even though they may not always represent the actual distances to the walls at present. Once per second, the robot checks if the distance in front of it is less than a specified threshold, if the threshold is not met, then the robot continues to walk straight forward. If the threshold is met, then there are two cases that can occur. The first case occurs if one or both of the current left and right sensor readings is less than the maximum sensor distance. In this case, a calculation is executed in order to realign the robot such that it is parallel with the walls of the hall (Figure 6).

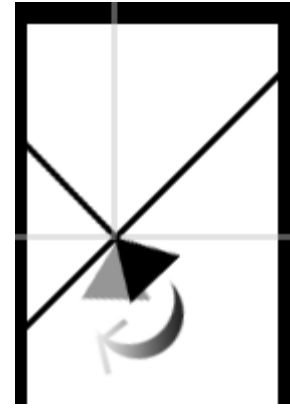


Figure 6: Normal Rotation

In the second case, both the left and the right sensor readings are equal to the maximum sensor distance, it is then assumed that the robot is heading straight towards a wall and a 90 degree rotation is executed in the proper direction based on whether the robot is supposed to be traveling clockwise or counter-clockwise (Figure 7.)

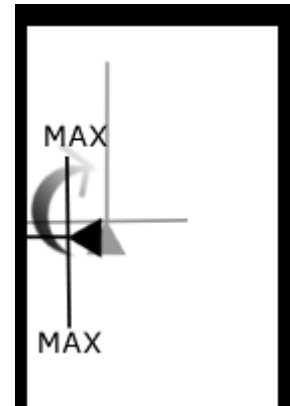


Figure 7: Rotation when left and right distance measurements are maximal

Visual Landmark Recognition System

Our vision visual detection of landmarks is based on the Tekkotsu vision system which functions in summary as follows. Frames from the camera are treated in YUV (Figure 9) space instead of RGB space (Figure 8) as humans are used to seeing them because by ignoring the Y channel, brightness variation can be largely ignored in color detection. Frames are then thresholded into segmented

into the four landmark colors and “unclassified.” Pixels of the same color are then joined together to form regions (Figure 10). The Tekkotsu vision system then picks out the largest region of each color present and makes information about the location (X and Y coordinates in the image) and dimensions of the object available to our programs.

Our landmark detection behavior maintains a list of the largest region of each color it has last seen. When all four colors are seen simultaneously it attempts to identify which color is in each corner by first identifying which two squares are in the top and bottom rows and in the left and right columns. A square is considered to be in the top row if its center Y coordinate in the image is higher than at least two other squares and to be in the left column if its center X coordinate is less than at least two other squares. Once the row and column of each square is determined it is trivial to determine the color in each corner.

To reduce erroneous landmark detections, we ignore a potential landmark when the color regions detected are too different in area, are too far apart or are not seen within one frame of each other.

Once the configuration of colors has been determined, it is matched against a stored list of landmarks to determine the robot’s location.

Unfortunately though light intensity is a relatively small problem, different light sources (e.g. florescent, incandescent, natural) cause the squares to appear different to the camera and the color thresholds must be painstakingly calibrated for each lighting condition. As the robot travels over its environment there are a variety of light

sources meaning that the colored regions are not always reliably detected.

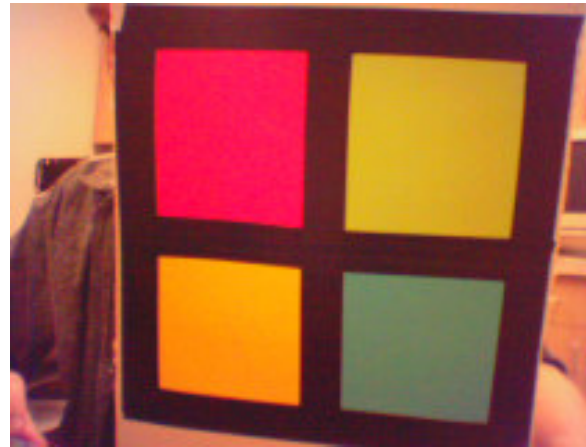


Figure 8: Vision frame in RGB space as human eye would see it.

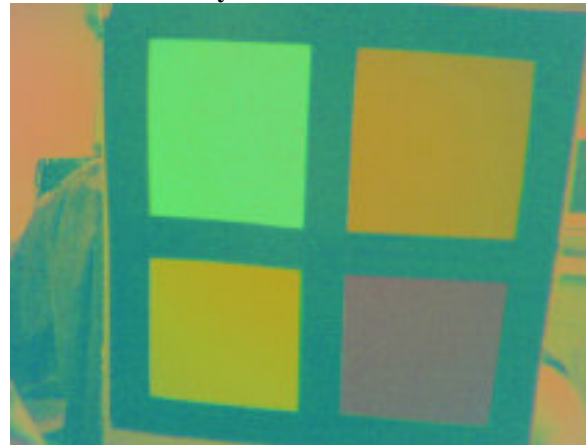


Figure 9: Vision frame in YUV space as robot sees it before processing.

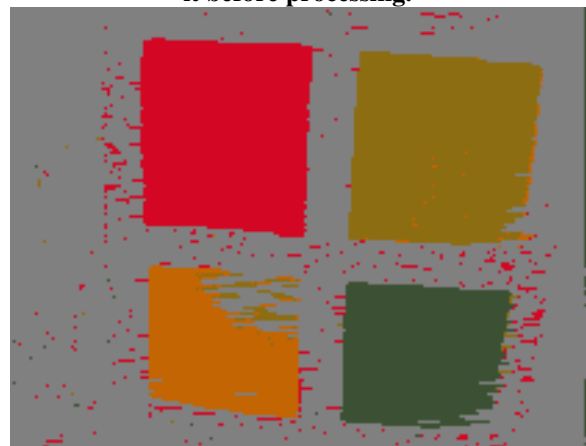


Figure 10: Color segmented image after processing. As you can see the landmark has become a set of four blobs of different colors.

Because of the above difficulties with the vision system, as the robot travels around the domain, frequently it will see something like figure 11 where only three of the colors have been detected. For a human it is very easy seeing this image to determine that the missing color is green, because yellow, red and blue are seen, and that it belongs in the lower right corner. Based on this idea we attempted to create a landmark detector which could similarly extract the configuration when only three colors were detected.

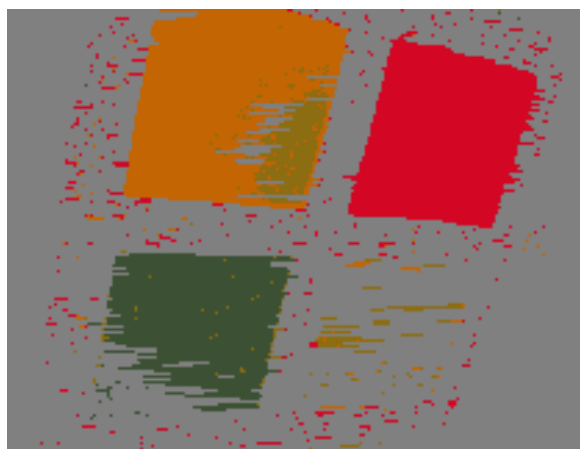


Figure 11: Segmented image where only three of the colored squares have been detected by the vision system. This condition occurred dismayingly often leading us to attempt to create a landmark detector which could extrapolate the missing square from the image.

During the limited time we had for our experiments we were unable to fully develop an algorithm to successfully perform this function. While determining the missing color is trivial, determining which corner each of the detected colors is in and thus which corner the missing color is in, proved difficult. Time permitting we would like to continue developing the landmark detector to make it more robust and easier to use in more environments.

Conclusion

Co-navigation of two robots in a one-dimensional domain is a task which can be addressed in a single term independent study. In the process of addressing this task we implemented inter-robot networking, hallway navigation and visual landmark recognition.

Future Work

There are two areas where we would like to expand this project:

- First improving the visual landmark detection system as described above.
- Additionally moving to a more complex domain would bring this project closer to real world applications. For instance moving to a two-dimensional domain or introducing obstacles to navigate around in the “one-dimensional” hallway.

References

- [1] “Tekkotsu: an open source project created & maintained at Carnegie Mellon University”
2006-06-06 <www.cs.cmu.edu/~tekkotsu/>