CSCI 445 – Group 2 Final Report

Gabriel Fuhrman

Kevin Greene

Raghavendran Venkatasubramaniyan

Josh Villbrandt

December 14, 2009

Miss Bianca was group 2's robot for the CSCI 445 final project. *Miss Bianca* was able to fully complete the search and rescue scenario. Although erroneous compass reading and varying battery voltages challenges our robot's movement, a strong particle filter was able to fully localize *Miss Bianca* after each move. Although this final project was very difficult, group 2 gained a lot of practical experience in robot design and programming, and the end result was very rewarding.

Introduction

For the CSCI 445 final project, each group must create a robot to autonomously complete a search and rescue mission in a "nuclear waste processing plant." Figure 1 shows the layout of the nuclear waste processing plant, which is actually constructed using eight-inch walls over an area of eight feet by twelve feet. As the scenario states, a nuclear core has become unstable and will meltdown in ten minutes. The goal of the robot is to remove the two victims that are trapped in the building and put the nuclear core in the containment room before the ten minutes expires. Another requirement is to take a picture of the victims and reactor core before moving them. Some of the challenges that this project incorporates are localizing the robot, moving the robot, identify objects, and manipulating objects.

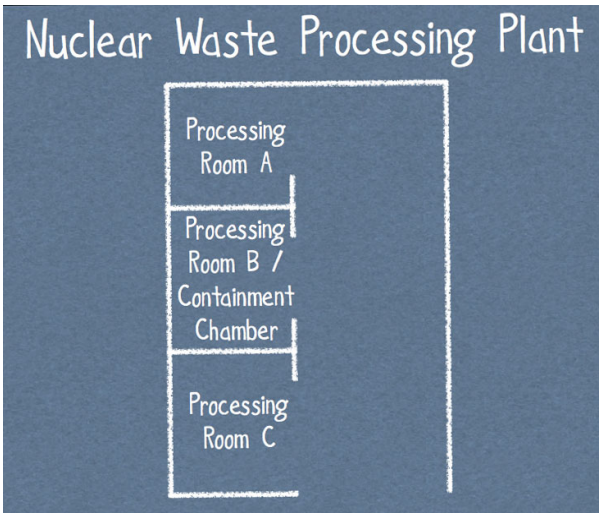

Figure 1: Processing Plant Layout

Procedure

To complete this mission, robots have to be able to reliably move around the processing plant and be able to manipulate objects. Our group built the frame of our robot using Lego. Two DC motors powered the wheels of the robot. The brain of our robot was a Gumbot board which used a Gumstix COM to run C++ code. To sense the environment, we installed a USB webcam, a compass, and a sonar module to the Gumbot. To get the best use of our sonar, we mounted

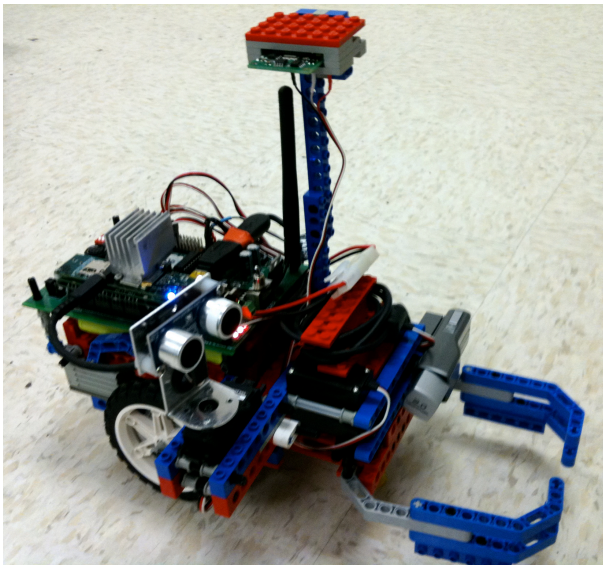

Figure 2: Group 2's robot, Miss Bianca

the sensor on a servo so that the sonar could measure with a range of 180°. And, to manipulate objects, a grabber was created using two additional servos. In the spirit of rescuing, our robot was named *Miss Bianca*, after a character from the classic animated movie, *The Rescuers*. Figure 2 shows a picture of *Miss Bianca* in her final iteration.

The success of this mission strongly depends on being able to localize the robot within the processing plant. Because of the inconsistent movements of the robot, attempts to move around the processing plant without being able to determine the robot's true location will surely fail. Our localization was accomplished by using a particle filter. In the particle filter, a number of simulated robots, or particles, are used to estimate the error in robot movement. By taking measurements with the sonar and compass and comparing those results to the expected value of each particle, the probability of each particle can be calculated. Then, the location of the robot can be assumed to be the location of the particle with the highest probability. For further accuracy, our particle filter resamples particles after each measurement. Appendix A and B go into more detail about how the probability of each particle is calculated as a result of sonar and compass measurements.

The next challenge that *Miss Bianca* faced was identifying objects. The objects to be identified were the victims, which were green blobs, and the reactor core, which was a red block of Legos. A picture of the objects is shown in Figure 3. To accomplish object recognition, images from the webcam were segmented into blobs against known colors in HSV space. If any blobs were found of at least a certain number of pixels, then the robot assumed that the largest blob was either the core or the victim, depending on the segmentation bias. From this point, the relative location of the object was calculated based on its position. By knowing the range of view of the camera and the horizontal position of the blob in the image, the relative angle of the object can be calculated. And, by knowing the

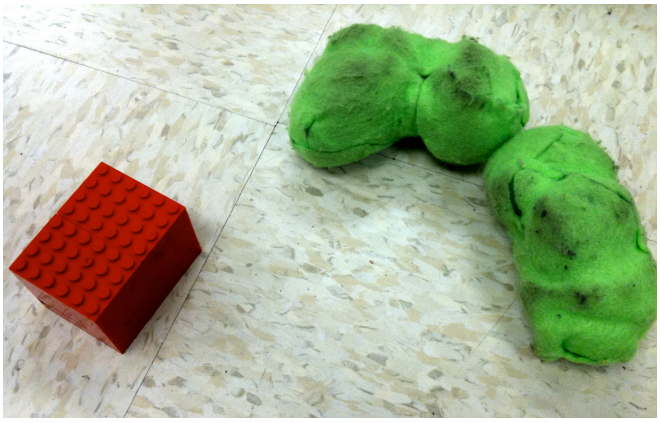

Figure 3: Processing Plant Objects

vertical position of the blob in the image, the distance to the object can be calculated. By knowing both the angle and the distance to an object, the robot is able to move towards the object and grab it.

With reliable localization and object manipulation, *Miss Bianca* still needs to know what to do. The overall search strategy starts by going to the core containment room and searching for an object. If the core is already in this room, Miss Bianca can immediately forget about all tasks related to the core, because the core is already in the required location. From that point, the robot would search in room A and then in room C to rescue victims. If there was a victim in the core containment room, the victim will be removed, and then the robot would search in room C and then in room A. This search order is the optimal search order for a robot that can only manipulate one object at a time. In each remaining room, *Miss Bianca* searches for an object and moves the object to the appropriate place. Since *Miss Bianca* can only hold one object at a time, she must move each victim outside of the processing plant before she can consider another task.

For the actual path planning to move from point to a point, an algorithm was created with a series of if-statements and some smart tricks to create a list of intermediate states. In general, if *Miss Bianca* is in a room and the goal state is outside of the room, the planner will add the center of the room, the entrance of the room, and the center of the hallway to the list. If the robot is in a cone that is close to the entrance of the room, the center of the room will actually not be added to the list. This makes movement more efficient, as the robot does not have to backtrack. Figure 4 shows a picture of this cone for room B. Then, if the goal state is within a different room, the planner adds the center of the hallway in front of the room, the room entrance, and the center of the room to the list. Again, the center of the hallway is not added if the

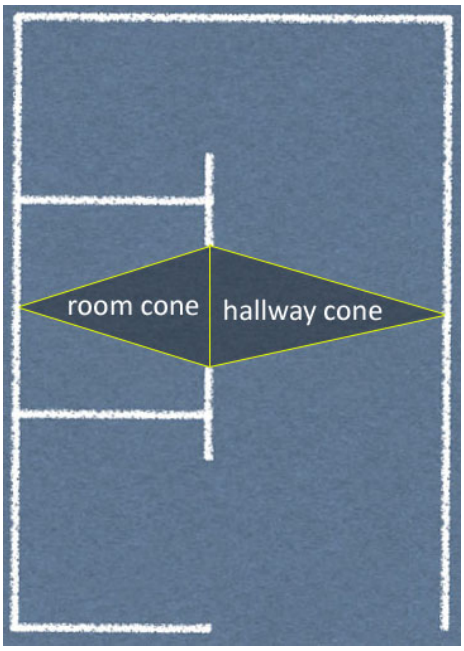

Figure 4: Path Planning Cones

previous state is within a cone in front of the room. This too is shown in Figure 4. Finally, the actual goal state is added to the list. Another trick that the planner does is making sure that the distance between successive intermediate states is less than a certain distance. This is an easy way to ensure that *Miss Bianca* always perfectly localizes. If an intermediate move is too long, more intermediate steps are added.

With a list of intermediate states in hand, *Miss Bianca* sequentially executes each move by turning to the next state, moving forward, and then measuring the surroundings with the sonar and compass to relocate. Turning is accomplished with a PID control class. Moving forward is accomplished by driving for a specific amount of time. Finally, sonar measurements are taken. By taking advantage of the fact that all walls are facing cardinal directions, *Miss Bianca* can ensure that she only ever measures distances in cardinal directions. This guarantees that all sonar beams will have diffuse reflections and will return the correct distance. (A specular reflection would cause the sonar to return the maximum value and would in turn throw the simulation off.) Since *Miss Bianca's* sonar can measure anything within 90° of her right side, there will always be at least two angles that are at cardinal directions. The actual distance is measured in each cardinal direction whose expected distance is within the limitations of the sonar. After figuring out these angles, an implementation of Bresenham's line algorithm was used to calculate the expected distance. This allowed *Miss Bianca* to get the expected distance of an arbitrary angle. With this data in hand, the particle filter can be updated, and the robot can be effectively localized.

Miss Bianca is also capable of localizing from any location in the processing plant. When starting from an unknown location, the first step *Miss Bianca* takes is reinitializing the simulated particles evenly across the map. Then, sonar measurements are taken in all four cardinal directions. If, at this point, the average position of the particles is in the hallway, the robot assumes complete localization. If this is not the case, then *Miss Bianca* assumes that she is in one of the three rooms. Since all three rooms look identical to the sonar, *Miss Bianca* has to drive out to fully localize. The first step in driving out of a room is taking a sonar measurement to the south so that *Miss Bianca* can figure out her vertical distance to the center of the room. After driving to the center of the room, *Miss Bianca* uses

the average position of the particles to figure out the horizontal distance to the center of the hallway. Once in the center of the hallway, sonar measurements are again taken in each cardinal direction. At this point, *Miss Bianca* is completely localized and the particle filter is reinitialized to the most likely state.

Results

After two runs, group 2 was successfully able to fully complete the scenario. During the first run, *Miss Bianca* was not able to retrieve the victim in room C. Although clearly identified by the image segmenter, *Miss Bianca* turned off course during the drive to pick up the object. Although this was a problem that had occurred before, it only happened about one in every ten trials. During her second run, *Miss Bianca* was able to overcome this problem and was able to complete the mission in about eight minutes. Figure 5 shows a sample output from the particle filter with the planned path overlaid on the image. The yellow dots are the simulated particles, and the red dot is the most likely particle. Figure 6 shows a sample output from the segmentation algorithm. A summary of constants that were used the second run is shown in Table 1.

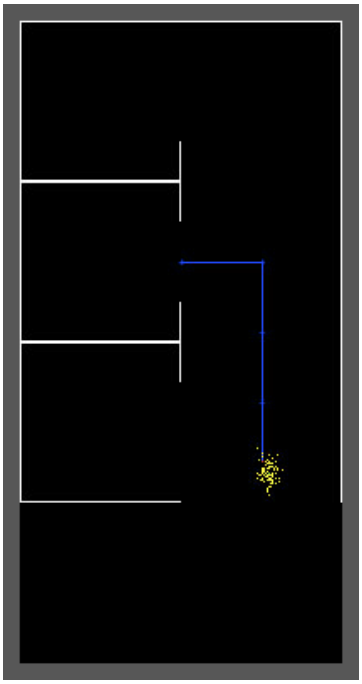

Figure 5: Particle Filter Output

Figure 6: Sample Segmentation Output – The first block is the raw camera output. The second block is the victim segmentation output. The third block is the core segmentation output.

Table 1: A summary of constants used in the final run.

PID proportional gain: PID derivative gain:	1.5 -0.5	PID integral gain: PID integral max:	0.4 60
Victim Hue Threshold: Victim Saturation Threshold: Victim Brightness Threshold:	105±10 50±50 128±128	Core Hue Threshold: Core Saturation Threshold: Core Brightness Threshold:	10±7 60±40 128±128
Simulated Particles: Sonar Standard Deviation: Compass Standard Deviation:	128 3.0 5.0	Straight Distance Std Dev: Straight Angle Std Dev: Turn Angle Std Dev:	4.0 4.0 5.0

Discussion

Miss Bianca was arguably one of the most consistent robots in the class. This is a direct result of a number of things that worked very well. For starters, the sonar sensors that were provided were extremely accurate. During isolated testing, the sonar sensor never read more than a half an inch off of the actual distance. This translates to only one pixel of error in our simulation, which used 24 pixels per foot. As a result of this, our localizing worked very well with the most likely state always being within three inches of the robot's actual location. Our PID algorithm was also very effective once the motors were sufficiently geared down. Tuning this PID algorithm did take time, however, and was continuously adjusted to overcome differences in battery voltage. Eventually a set of gains was found that never had to be adjusted. These gains are shown in Table 1. The movement of our robot was also very consistent. This was a result of a very simple and sturdy design. Since *Miss Bianca* was smaller than all of the other robots, she was also less prone to getting stuck on a wall. Finally, our segmentation was always very fast and accurate. From Figure 6, one can see that there was very little noise in our segmenter. This was due in large part to our image segmenter working in HSV space rather than RGB space.

There were also a number of things that only worked moderately well. This includes compass readings, which required a first-degree polynomial just to get within 7° of any angle. Even then, the magnetic field in the lab room caused compass readings to be up to 45° off from one location to the next. Another constant problem was varying battery voltage, which caused physical constants, such as distance traveled per unit of time, to vary along with it. The battery turned out not to be such a big problem, because the particle filter was always able to compensate for movement error. A final problem was moving to pick up an object. Although the absolute angle of an object after the initial segmentation was always very accurate, turning towards the object while moving towards the object was inconsistent. This is either a result of a poorly calculated angle or a poorly executed turn. Not enough effort was put in to this problem to narrow down the problem any further, since *Miss Bianca* was able to collect both victims during the second run.

Overall, *Miss Bianca* was a very successful robot. She was the first to fully complete the scenario and the most reliable at completing the scenario. *Miss Bianca* was also the only robot to draw her intended path on the simulation output, the only robot capable of using sonar measurements from non-cardinal directions to effect the probability of simulated particles, and the only robot to fully localize from any place within the processing plant. The challenges that we faced during the construction of *Miss Bianca* gave us first hand experience with the challenges of building and programming a robot. After weeks of hard work, completing the scenario was very rewarding.

Appendix

A. Probability of a particle as a result of a sonar measurement.

The pseudo code below is applied to every particle in the simulation after a sonar measurement has been taken.

```
expectedMeasurement = expectedDistance(particles[i].state, measurementDirection, false);
probReadingGivenLocation = normal(measurement, expectedMeasurement, sonarStdDev);
probCurrentLocation = particles[i].probability;
particles[i].probability = probReadingGivenLocation * probLocation;
```

B. Probability of a particle as a result of a compass measurement.

The pseudo code below is applied to every particle in the simulation after a compass measurement has been taken.

```
expectedMeasurement = particles[i].state.angle;
probReadingGivenLocation = normal(measurement, expectedMeasurement, compassStdDev);
probCurrentLocation = particles[i].probability;
particles[i].probability = probReadingGivenLocation * probLocation;
```