

# Distributed Intelligence in Autonomous Robotics

## Assignment #3

**Out: Thursday, February 13, 2003**

**Due: Tuesday, February 25, 2003**

### Randomized Search Evaluation

In this assignment, we will explore the effectiveness of a distributed, randomized search strategy for target detection as a function of the probability of single pass target detection. Several experiments will be run with 6 robots. These experiments will vary the single pass target detection probability, and data will be collected that measures the percentage of targets detected over time. This data will be graphed and compared to some of Gage's results in paper #7.

#### 1. Randomized Search Program

For the basic randomized search algorithm, use a revised version of the `sobounce.c` example program that was provided to you in Assignment #1. This program uses sonar to cause the robot to wander through the environment, avoiding obstacles. (When you use this code, be sure to remove the calls to `server_is_running()`, since this interferes with multi-robot runs in the simulator.) This code will be revised as described below. In this discussion, the revised code will be called the `Randomized_Search` program. At a high level, your `Randomized_Search` code will do the following:

- Begin in a wait state until a starting position is received from the `Data_Collect` program (described below).
- Once the starting position is received from the `Data_Collect` program, the `Randomized_Search` program places the robot in the assigned starting position and returns to a wait state.
- The `Randomized_Search` program waits until it receives a "probability of detection" parameter from the `Data_Collect` program. This parameter indicates the probability of detecting a target on a single pass. This will be used as described below to determine whether or not a robot has detected a target.
- The `Randomized_Search` program waits until it receives a "Begin Search" message from the `Data_Collect` program. At this time, it begins the wandering/searching process.
- Whenever the robot detects a target, it communicates this information (including the position of the target) to the `Data_Collect` program.
- The `Randomized_Search` program continues until it receives a "Stop Search" message from the `Data_Collect` program. At this time, the program exits.

This process is described further below.

#### 2. Data collection program

To collect experimental data and control the initialization, starting, and stopping of the experiment, write a program called `Data_Collect` that does the following:

- At the beginning of the experiment, `Data_Collect` generates non-overlapping starting positions for the 6 robots. These starting positions should use the same algorithm that you implemented for Assignment #2; that is, positions are randomly selected from a uniform distribution within a circle centered at (0,0). **For these experiments, the circle should have a radius of 700.** For this assignment, all starting positions should be generated by the `Data_Collect` program to ensure that the robot starting positions do not overlap (taking into account the size of the robot). Once 6 non-overlapping starting positions have been generated, the `Data_Collect` program assigns each robot to a position and broadcasts these assigned positions to the robots (using the communications server).

- The Data\_Collect program sends a message to all the robots indicating the probability of detecting a target on a single pass. For these experiments, different experiments will be run that assign these probabilities to one of: 0, 0.2, 0.4, 0.6, 0.8, and 1. All robots should be using the same probabilities of detection in a given run.
- The Data\_Collect program should generate a time stamp for the beginning of the experiment, and then immediately communicate a “Begin Search” message to all 6 robots’ Randomized\_Search codes.
- The primary purpose of the Data\_Collect program is to collect data on the cumulative percentage of targets detected by the distributed robot team as a function of elapsed time from the start of the experiment. The Data\_Collect program knows in advance the location and number of all targets. After the experiment begins, the Data\_Collect program should continually check for messages from the robots indicating a target has been detected. The Data\_Collect program checks to see if this is a newly detected target (i.e., that has never been previously detected by any other robot). If so, then it updates its records of the cumulative percentage of targets detected as a function of elapsed time. Note that the robot team does not get credit for detecting targets that have already been previously detected (by any robot).

For this data collection, the “time bucket” (i.e., discretization of time) should be in 5-second increments. So, at the beginning of the experiment at time  $t=0$ , the percentage of total targets detected will be 0. At time  $t=5$  seconds, if any new targets have been detected, then the percentage of targets detected at that time will go up, and so forth. This percentage will steadily rise until at some point in the experiment, all targets have been detected (100% detection).

- Once a desired percentage of targets has been detected (ideally, 100%), then the Data\_Collect program recognizes this fact and sends a “Stop Search” message to the robots, at which time the robot programs exit. The Data\_Collect program then outputs the collected data to a file, which is the cumulative percentage of total targets detected from the start of the mission to the end of the mission, in 5 second increments, for the current probability of detection level.

### **3. Target locations**

In these experiments, the experimental setup will be a square area bounded by walls. Targets will be located on a regular grid every 1000 units. That is, there will be targets located at (0,0), (0, 1000), (1000, 0), (1000, 1000), (1000, 2000), etc., throughout the entire experimental area. You do not have to do anything to mark these target locations. This information will just be known to the Data\_Collect program. Your Randomized\_Search code SHOULD NOT take advantage of this knowledge to control the robot’s motions to increase the likelihood of target detection.

### **4. Probabilistic Target Detection**

Add a capability to your Randomized\_Search program that enables a probabilistic detection of targets. As stated earlier, the Randomized\_Search program will receive the probability of detection parameter from the Data\_Collect program at the start of the experiment. For each loop through your Randomized\_Search code, the algorithm should check to see if the robot is within a distance of 200 of a target. Remember, a target is located at every (x,y) position that is a multiple of 1000. If the robot is within this distance of 200 of one of these positions, then a random number generator should be used to determine if the robot actually detects the target, according to the assigned probability of detection. (For instance, one way of doing this is to use a random number generator that returns a value between 0 and 1. If the assigned probability of detection is 0.2, then the robot detects the nearby target only if the random number generator returns a value of 0.2 or less.) If the target is detected, then the robot sends a message indicating the position of the detected target the Data\_Collect program. The Randomized\_Search code should also keep a memory of the last target that was within a distance of 200, and only check 1 time for detection of each new target position. This is important, in order to remain true to the concept of probability of “single pass” detection. So, for example, if a robot is nearest to a target at position (3000, 1000), then it only calls the random number generator one time as long as the closest target position is (3000, 1000). If the robot comes back later to be closest again to the

target at (3000, 1000) (after being closest to some other target in the meantime), then it can call the random number generator another time for that pass to see if it is detected.

### **5. Test environment**

Generate a test environment (i.e., map in the Nserver) that is a square of size 10,500 x 10,500, centered at (0,0). This test environment will therefore have 121 targets (11 x 11).

### **6. RUN THE FOLLOWING EXPERIMENTS:**

For all of these experiments, use the test environment from #5 above, with 6 robots. The robot starting positions should be randomly generated as described above within a circle of radius 700, centered at (0,0). The robot sensing range of targets is 200, subject to the probability of detection parameter.

- Run 5 experiments, assigning the probability of target detection to one of {0.2, 0.4, 0.6, 0.8, 1.0}.
- For each experiment, collect data that shows the cumulative number of targets detected as a function of time, from the start of the mission until 100% of the targets are detected, in 5-second increments.

### **7. GRAPH RESULTS:**

Using your favorite graphical plotting package, generate one graph as follows:

- x-axis is time (in 5 second increments)
- y-axis is cumulative percentage of targets detected
- 5 curves shown (one for each of the probability values listed above), with an interpolating curve between each of the data points for a given detection probability
- Legend that clearly labels each curve according to its probability value

This graph must be generated and plotted electronically, not by hand. The graph should have a title, the axes should be labeled, and the 5 curves should be easily distinguishable.

### **TURN IN THE FOLLOWING (A-G):**

- A. Your graph from part 7 above.
- B. A table showing the raw data from which your graph was generated, clearly indicating which set of data corresponds to which experiment.
- C. A discussion (on the order of 1 paragraph) of what the results in your graph show. This discussion must be typed, not hand-written.
- D. Compare your results for probability of detection = 0.8 to the lower curve of Figure 2 in the Gage paper (#7). Write a 1 paragraph discussion comparing and contrasting your results with those of Gage. If the results are similar, discuss why that should be the case. If the results are different, discuss why your results are different. This discussion must be typed, not hand-written.
- E. A typed discussion of any issues you had to deal with in order to make your code work properly.
- F. A hardcopy of your code (both programs), fully documented according to the instructions from Assignment #2.
- G. Create a tar or zip file of your code, including all files needed to compile your code in an empty directory (e.g., Nclient.o, Nclient.h, inter\_robot\_communications., etc.). You do not have to include the communication\_server.c code. **Be sure to test the unpacking and recompilation of your code in an empty directory, to ensure that you have included all necessary files for code compilation.** Email a hardcopy of your code package to [parker@cs.utk.edu](mailto:parker@cs.utk.edu), naming your code “yourlastname-3.tar” or “yourlastname-3.zip”.