Proceedings of 2002 JUSFA:
2002 Japan-U.S.A Symposium on Flexible Automation
July 15-17,2002 Hiroshima, JAPAN

U-007

DEVELOPMENT OF AN INTERACTIVE SIMULATION WITH REAL-TIME ROBOTS FOR SEARCH AND RESCUE

D. A. Campos, Elizabethtown College
camposda@etown.edu

J. T. Wunderlich, Elizabethtown College
wunderjt@etown.edu

ABSTRACT

This research involves the use of cooperative mobile robots for use in search and rescue. A two-part process uses the analysis from a concurrent simulation that directs actions of surveying robots in the field while modeling the robots' environment. Expanding the simulation part of the network leaves room for study of different scenarios.

INTRODUCTION

Three mobile robots have been built and programmed for search and rescue. The general problem is the development of cooperative mobile robots commencing a search and rescue effort through the use of an interactive simulation. This paper addresses the issue of having variable data monitored by a single simulation and affirming the possibility of search and rescue given the constraints of the relatively inexpensive mobile robots. Communication between robots can be monitored with the use of a LEGO Mindstorm IR tower. After establishing the link with the robots a closed-loop system was developed to produce findings through the surveying robots in the field.

The main robot used is the Scout that gathers the fundamental data to be processed in the simulation once returned via Datalog. All of the mobile robots are programmed using the Not-Quite-C (NQC) programming language described in [1], which Baum shows is almost like the C language. The RCX brick has a default language and interface, RCX code, but is aimed at young consumers [4]. The NQC language is much more flexible because of its ability to use data structures and not as limiting as the RCX language [1,2,3].

The Datalog is a useful function available through the RCX that records data into a matrix that can be fed back to the CPU. The following lines of code would designate creating the Datalog matrix in the Scout, and adding the value of one of three of its internal timers:

```
CreateDatalog(1000);
AddToDatalog(Timer(0));
```

The results of the Datalog return to the CPU in a file with corresponding variable numbers and values.

The MATLAB simulation can be used to interpret the encoded data. Such software was chosen due to the ease of creating the simulation environment and manipulability of mathematical modeling. The simulation window in Figure 1 shows the anticipated path taken by the robot from the data.

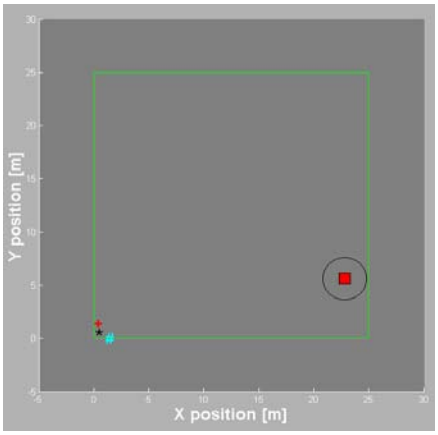


Figure 1: Simulation Output Window. Grey designates unknown area, robots shown: scout (black), medic (blue), fire-suppressant (red), and a light source (red).

There are two different approaches to the study: one measures only time and direction, the other incorporates displacement. Without the proper inputs from the scout, we are limited to knowing only the direction of the scout and how long it moved in that direction. A large assumption, using this method, is that the robot is constantly moving. We are assured that there have been no obstacles in the way because of the bump switches on the front of the scout that monitors the presence of walls or objects in the environment. However, as in most mobile robot studies, the event of slippage cannot be monitored without incorporating relative movement of the robot to its surroundings [5].

The study was done using only the internal timer, and bump sensors on the RCX block. To have full comprehension of the terrain using only three sensors (possibly four in the future) goes beyond the scope of this paper. In order to assure that we are not just "spinning our wheels", the study is recorded visually (via camera) to compare actual results. However, in the real world, we must do without the camera, obviously, incorporating displacement and tactics to counter slipping for a more accurate localization of objects in an unknown or untraversable terrain. A third "benchmark" is considered in the study for a theoretical analysis of what should happen given a reproducible testing environment (e.g. a pen with positioned objects).

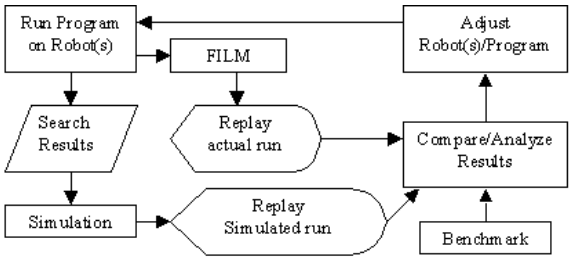


Figure 2: Overview of Entire Study Flowchart.

In order to gather and interpret the incoming data from the recording Datalog, the Scout must return to the CPU interface (the IR tower) and report its findings. A closed loop system is used to iterate through the Datalog in order to simulate the actions as they occurred. These animations are drawn on the map, which the simulation is keeping track of. Figure 3 illustrates the use of such a system. A large assumption in the study is that the simulation has unlimited memory and time is always flexible.

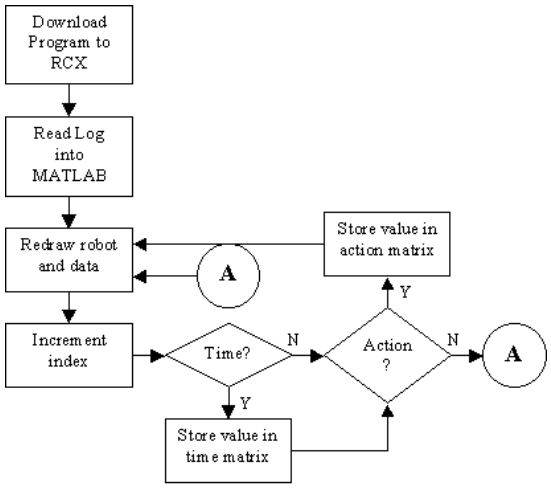


Figure 3: Data Process Flow Chart. Simulation iterates through Datalog to filter the illustration onto the map.

The problem of the dimensionless value of distance in the study has been replaced with a general tested velocity of 7.7sec/m, which will be used to convert time to distance in the maps. An optical encoder could not be used since the three sensor ports of the RCX block were in use. The present organization is the least complex and satisfies the nature of this study. Future test runs will use an optical decoder to measure distance by wheel rotations.

Previous studies using the LEGO Mindstorms have brought results showing capabilities of communication among each other [2]. After gaining knowledge of the environment, the simulation can inform the other mobile robots of their tasks. To complete this, task the "-msg" and "SendMessage()" functions are used throughout the search process (perhaps rescue).

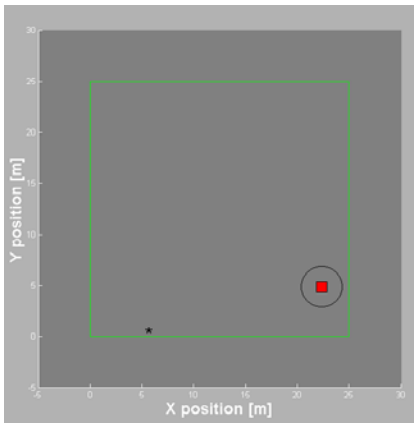
Figure 4: IR Tower and Scout Robot. The robot must face the tower in order to communicate with the Simulation.

Two other mobile robots have been built and are fully functional clones of the Scout; but have been programmed to react to the findings of either the "fire" or the "human." The necessary condition for their intervention varies among sensor types on the robots. Using a candle alone as a light source merely demonstrates the ability to use such a system.

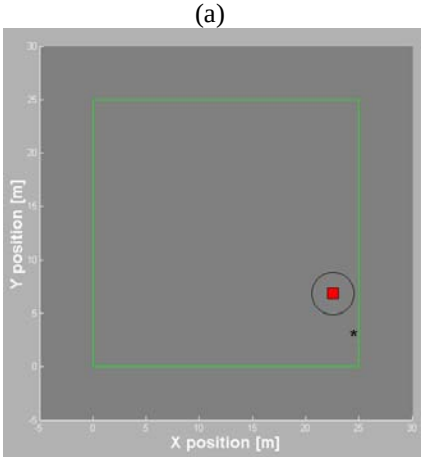
TESTING

The scenario in place is having the Scout in a situation where it finds a light source after bumping into one wall. After witnessing the robot's search, the data is sent back to the simulation. The Datalog returned is a readable set of three columns of raw data. By parsing the information, the information could be processed through the output filters of the previous flowcharts.

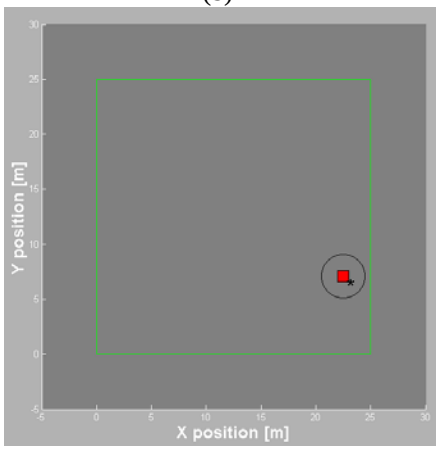
While the CPU was able to send the message that would alert the other mobile robots, it was unclear as to how much time was needed to keep the other robots active. Therefore the robots were kept on a sleeping algorithm to wait for a message from the Scout whether they were needed. An example of what occurred can be seen in Figure 5.



(a)



(b)



(c)

Figure 5: (a) Start of search by Scout. Scout is asterisk on. (b) Scout bumps wall but makes the predefined turn to avoid the obstacle. (c) Scout finds light source.

Further testing showed that the light sensors on the robots have a range to detect the incoming light source. Data from the Datalog can be read into a simulation in this manner to display results. The actual timing output in the simulation was not as exact as the actual robot. This may be due to the RCX's internal timer's constraint of only outputting time in 100ms resolution. A newer version of this hardware allows better resolution thus better results. Furthermore, by adding the Datalog element into the other two robots, we may be able to keep track of the actions of all of the robots. This would mean having multiple Scouts and multiple emergency robots.

CONCLUSION

From the results and tests we are positive that using this model of a coordinated effort for search and rescue is emerging. Using the MATLAB simulation along with the Datalog in the RCX Scout allows many possibilities. The nature of the study was made on the basis of limited time and finances. Further testing and redesign is always possible, but to do so would change the basic structure of our closed-loop system. Several instances in the study begged the question of whether nonlinear anomalies in the system would have an overall effect on this type of venture. Since the robot uses a limited power supply (it uses NiMH batteries), power consumption was always an issue. Furthermore, friction of surfaces and eventual gathering of dust (if not a monitored environment) will affect the results. Future considerations may be to use neural networks to initiate map learning or development of occupancy maps where the ability to maneuver relates to space known to be previously searched.

REFERENCES

[1] Baum,D., *Definitive guide to Lego Mindstorms (and Not-Quite-C)*. Emeryville, CA: Apress, 2000.

[2] Pittinger, B., Drill, T., Glasby, W., and Shank, K., Simulation and real-time control of DewEbot: an Elizabethtown College mobile robot, CS/ENGR 344 semester project report, Elizabethtown College, Fall 2000.

[3] Zabriskie, K., and Drill, T., RCX, Not Quite C, and Data Logging: an Elizabethtown College study to compare two programming languages, CS/ENGR 344 semester project report, Elizabethtown College, Fall 2000.

[4] Wallich, P., Mindstorms Not Just a Kid's Toy, *IEEE Spectrum* (September 2001) 52-57.

[5] Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M., and Schimdt, T., Map Learning and High-Speed Navigation in RHINO. In *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, Kortenkamp, D., Bonasso, R.P., and Murphy, R., eds., 21-52. Cambridge, Mass.: The MIT Press, 1998.