An Initial Evaluation of Approaches to Building Entry for Large Robot Teams

Zheng Ma, Peiju Lee, Ying Xu and Mike Lewis
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 152260, USA
{zhm5, pel30, xuy8}@pitt.edu and ml@sis.pitt.edu

Paul Scerri
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
pscerri@cs.cmu.edu

Abstract — Teams of robots for search and rescue in dangerous building environments are emerging as a potential approach to reducing risk to rescuers and improving rescue rate. However, one aspect of the use of such teams that is often overlooked is the initial entry of the robots into the building. If many robots enter the building from the same entrance, it will often be the case that the robots interfere with each other and dramatically slow down the initial phase of the exploration. In this paper, we evaluate several different approaches to over-coming this initial congestion and identify heuristics that allow the robots to most quickly clear the entry area and begin their actual mission. Our results showed that unless the exits to the initial entry area are very small, the most effective technique is for every robot to simply move in the direction of the most open space.

Keywords: *Human-robot coordination, large teams, distributed coordination, autonomous path planning*

I. Introduction

Teams of robots offer an promising new way to perform search and rescue in environments where it is dangerous or difficult for human rescuers to go in. For example, if there is a fire or chemical spill or biological attack in a large building, a team of robots could be deployed to find victims, look for sources of problems and detect structural issues. One key to the effectiveness of any rescue mission is how quickly the team can deploy and search the building. Typically, the robots will enter the building at one or a small number of points and need to spread out from there. For example, the whole robot team might be sent in through a window into the only room that is accessible to the human rescue team. This will potentially create significant initial congestion for the robots, especially if there are only a small number of exits from the room to the rest of the building. An example of this is shown in Figure 1. One aspect robotic rescue response that has received little attention in the literature is how a team of robots can quickly overcome this initial congestion and spread out sufficiently that they can begin their mission.

Although the problem of overcoming initial congestion has been noticed in at least two well known multi-robot systems[1], [2], there are few published solutions. In MARTHA a token-based approach was used, that allowed one, or a very small number, of robots to move one at a time[1], hence slowing the overall efficiency. Some robotic swarm approaches initially appear relevant to this problem, but are

Fig. 1. Example showing congestion in USARSim

usually focused on keeping the swarm together, not spreading a team out. In any case, swarms are often relatively inefficient around tight openings, like doorways. Potential fields and other physics inspired models also initially appear to show some promise, however in a congested room, spreading out might not be possible until the robots have gotten out of the room they are in. Thus, the potential field approaches are ineffective until the initial congestion is cleared. Notice also that clearing the congestion will be very difficult and time-consuming for even the most skilled human operators, because the local viewpoints of the robots provide poor information for working out which robot should move where to clear the congestion. Thus, while we do not believe this problem represents a fundamental roadblock to robot teams, we believe that there is an absence of good solutions to the problem and that a good solution will be key to the practical effectiveness of robot teams for urban search and rescue.

The main technical challenge for the robots is that often their main sensing modalities will not work well when there are many intelligent moving entities close by. For example, laser scanners are not be capable of making useful maps when most of the scan is interfered by moving robots between the

scanner and actual environment features. Even if the map is known to the robots, locating themselves on the map can be infeasible if GPS is not available, as is typically the case indoors, because other robots obscure features required for determining locations. In some cases, it will be possible for the robots to distinguish between the features of the environment and other robots in the environment, e.g., with vision or sequences of laser scan data to subtract moving objects, but this can be difficult and time consuming. Moreover, unless the robots have some scheme to visually recognize one another, they may not know *which* robot is blocking their way, even when they can determine that is another robot. Thus, for many types of robots, the only useful sensor data available to them for making decisions will be current distances to obstacles, robots or environment features, in each direction around the robot. Effective decision-making under these circumstances is challenging.

In this paper, we present four promising approaches to clearing the initial congestion faced by a team of robots when first entering a building. The four presented approaches were the most promising of a larger set of approaches we looked at. None of the approaches relies on communication, requires knowledge of a map, requires being able to distinguish robots from the environment nor requires any special sensing capabilities. Thus, these four approaches will apply to most robots in most environments. The key to each of the approaches is to get as many robots moving at once as possible, while getting them to move in a way that opens up the space rather than blocking it up. One of the algorithms requires the robot to get something to their left and then move forward. This results in something like a swarming behavior and allows most of the robots to move forward at once. Another two are variations on trying to move in a certain direction until being stopped and then changing directions. The fourth approach requires the robot to find the direction with the most open space and move forward into that space, similar to the more sophisticated "move-to-the-frontier" exploration algorithms [10]. It is also the only algorithm in the set to require sensing beyond the ability to detect collisions.

We developed an abstracted simulation environment for doing systematic evaluation of the different algorithms. The simulator allowed factors such as the number of robots, the number of exits and the amount of clutter in the room to be varied. Results showed that the algorithm requiring robots to move to the biggest open space was the most effective at getting the robots out of the room. The wall finding and following algorithm was next best. Most experimental parameters impacted results in predictable way, e.g., more doors allowed the robots to clear the room more quickly. Surprisingly, the initial amount of congestion of the robots did not impact the time to clear the room. As the number of robots was increased, the performance of the wall following algorithm decreased dramatically, until it was no better than the move-until-hit algorithms, suggesting that for very large robot teams this approach may lose its effectiveness.

II. Problem Model

The initial aim of this work was to identify algorithms that may be worth evaluating on physical robots, hence the focus here is to capture salient features of the environment so that the evaluation at least qualitatively matches performance in physical robots.

The robots are modeled as square robots, about 30cm by 30cm, approximately the same size (though differently shaped) to the iCreate robots we intend to eventually test the algorithms on. The robots are modeled as homogeneous and capable of moving in any direction with no difference in effort. The environment is about 6m x 6m and filled with rectangular obstacles. The number and size of the doors in the room is varied systematically in the experiments. The location of the doors is not a priori known to the robots. A screen shot of the simulation environment is shown in Figure 2. The parameter of the experiment is 50 robots, 1 door with width 0.9m and about 6 medium size obstacles in the initial entry.

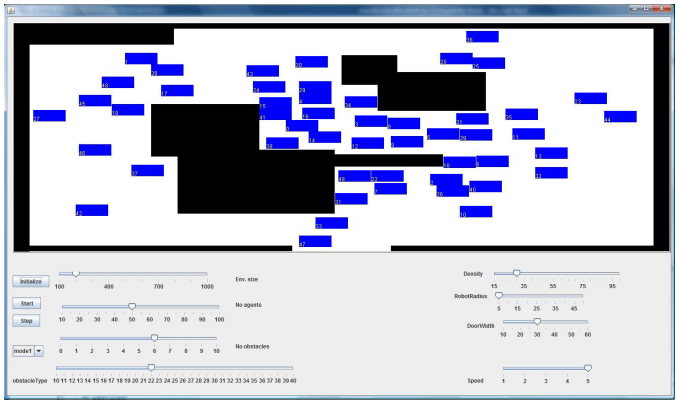

Fig. 2. Screen shot of the simulation environment

The robots can sense distance to the nearest obstacle accurately at the eight compass points around it. They cannot distinguish robots from features of the environment with their sensors. This model approximates a very simple laser scanner or an IR or Sonar ring.

In this initial work, we assume no communication is available to the robots. In general, the congestion would be both a benefit and problem for the robots. The benefit derives from all the robots being in communication range of one another, which allows any robot to communicate reliably with any other. The problem is that wireless communication is a shared medium, effectively dramatically reducing the available bandwidth per robot. For large teams, we anticipate that managing the shared medium across all the robots will be difficult and the value of communication low, because so little can be done per robot.

A. Algorithms

We have identified four simple algorithms that might plausibly solve the congestion problem without requiring localization, communication, or anything beyond rudimentary sensing. The algorithms are similar in combining extremely simple

plans of movement with a rescue routine that protects against the plan's particular path states.

Algorithm 1: Wall Following (left hand rule)
(1) **if** $leftSide$ is $true$
(2) **if** $turingAround$ is $true$
(3) findWall()
(4) **else**
(5) turnleft()
(6) **else if** $foward$ is $true$
(7) moveforward()
(8) **else** turnright()

Wall Following can use either a left hand rule or right hand rule. In the left hand rule, used by our algorithm, the robot follows the wall to its left. Notice that the robot cannot distinguish walls from any other object in the environment, hence "wall following" actually means having something close to the left. Wall following uses a greedy algorithm in which the robot moves to its left if possible, if not, it moves forward otherwise it moves to its right. At its initial stage if a robot is not adjacent to a wall it will turn left endlessly so a routine is needed to check for spinning behavior and call findWall() if it determines it is spinning. findWall() moves the robot in random directions until an obstacle (hopefully a wall) is found and wall-following behavior can take over.

Algorithm 2: Keep Going Until Hit and Algorithm 3: Change Direction When Stuck combine two simple plans, labeled A and B, in different ways. While the resulting behavior is strikingly different from Wall Following the instructions are very similar.

Plan A: counterclockwise move
(1) **if** $forward$ is true
(2 moveForward()
(3) **else if** $left$ is true
(4) moveLeft()
(5) **else if** $backward$ is true
(6) $hit \leftarrow$ true
(7) moveBackward()
(8) **else if** $right$ is true
(9) $hit \leftarrow$ true
(10) moveRight()

Plan B: clockwise move
(1) **if** $forward$ is true
(2 moveForward()
(3) **else if** $right$ is true
(4) moveRight()
(5) **else if** $backward$ is true
(6) $hit \leftarrow$ true
(7) moveBackward()
(8) **else if** $left$ is true
(9) $hit \leftarrow$ true
(10) moveLeft ()

Move plan A prescribes counterclockwise movement. The robot first tries to move forward. If blocked, it tries to move left. If blocked again the robot is considered to have hit an obstacle. The robot then tries to move backward followed by a move to the right if backward failed. Move plan B is similar except its direction sequence is clockwise: forward, right, backward and left.

Algorithm 2: keep going until hit
(1) **if** hit is false
(2) moveOn() [1]
(3) **if** hit is true
(4) randomDirectionSelect()
(5) randomSelect(planA(),planB())

In algorithm 2, Keep Going Until Hit, the robot is randomly assigned either plan A or B. For example, if a robot starts with plan A (counterclockwise), it tries to move forward then left. If it cannot move in either of these directions, a hit is declared. The robot then changes its direction randomly and selects at random between plans A or B.

Algorithm 3: change direction when stuck
(1) moveOn()
(2) **if** $stuck$ is true
(3) randomDirectionSelect()
(4) randomSelect(planA(),planB())

Algorithm 3, Change Direction When Stuck, is similar to algorithm 2 except it changes plan on determining a robot to be stuck rather than merely hitting an obstacle. Declaring a robot stuck requires that the robot has not moved further than 3 steps in any direction after attempting 30 moves in the algorithm. When determined stuck the robot changes direction and plan.

Algorithm No.4 :Move along longest path
(1) sortLaserScannerData();
(2) direction $\leftarrow$ directionWithLongestDistance();
(3) move(direction);

Algorithm 4 is called Move Along Longest Path because the robot moves in the direction in which it detects the longest open path using a laser scanner or similar sensor. First the robot sorts sensor data to find the longest open path. Next the robot moves to that direction. A problem for this algorithm is that the robot can become trapped equidistant between two obstacles, "vibrating" as the direction of the longest path reverses with each movement. The solution is that when vibration is detected, the robot moves in a random direction for (1-30) random steps. Notice that this behavior depends on a sensor that can detect obstacles (or lack of obstacles) at a significant distance from the robot, so it may not be useful

[1] moveOn() uses one of the plans A or B

for sensors such as IR or sonar.

III. Experiments

A. Metrics

To evaluate the performance of the algorithms, we have chosen time i.e., how long a robot takes to move out of the room, as the basic evaluation criteria. Time is measured in discrete "steps" of the simulation with each robot capable of moving a constant distance at a step. To find a metric of performance that best characterized the effectiveness of dispersing a robotic team, we looked at three options: median; maximum; and average. Median is the number of steps after which half of the robots had left the initial room. Intuitively, median captures the point at which the room is only half as congested as it started and the team should be on their way to starting exploration. However, median does not benefit algorithms that get some robots out of the room very quickly or struggle to get the last few out. Max is the number of steps until 90% of robots have escaped the room. 90% was used instead of 100% to avoid some distortions due to a single robot getting completely stuck or lost, since that behavior is likely independent of the congestion clearing algorithm. Intuitively, the Max metric focuses on clearing the room of the robots but does not reward algorithms that get some robots moving quickly. Avg is the average number of steps required for all robots to escape the room. Avg turned out not to be useful because when robots became stuck in the room and could not escape within a reasonable period (10,000 steps), Avg. was undefined. Even when most of the robots did eventually leave and others were ignored (as with Max), the Avg metric was so biased by the time the last few robots left the room it was not helpful in evaluating the performance of the algorithms with respect to dealing with the congestion. We plotted all metrics for each of the experiments, but below show only the results for Median. We believe it most accurately reflected the algorithms' performance dealing with the congestion and was least impacted by quirky performance by individual robots.

B. Results

The aim of this initial work was to find which algorithms work better than others so that we know which algorithms to implement for our physical robots in high fidelity simulation environments. Our key experiment was to vary as many salient parameters as feasible and look at performance over each of the parameterizations. Specifically, we varied the number of robots, number of exits to the room, the initial clustering of the robots(density), the size of the doors and the nature of the obstacles.

To evaluate the 4 candidate algorithms, we ran 20 repeated experiments under each parameter combination. The results are shown in Figure 3-Figure 7. Robots were initially distributed in a Gaussian way, with the "sigma" of the distribution varied to vary the initial congestion. Values of 0.01, 0.03, 0.07 were used below representing density of robots at initial stage. Obstacles were always rectangular but different types of environment were created by varying the size and number of obstacles. Below we represent the type of obstacle in an environment with $n-m$, where n is the number of obstacles in the environment, 1 for few, 3 for many and m is the size of the obstacles, 1 for small, 3 for large. For example, $1-1$ means few, small obstacles.

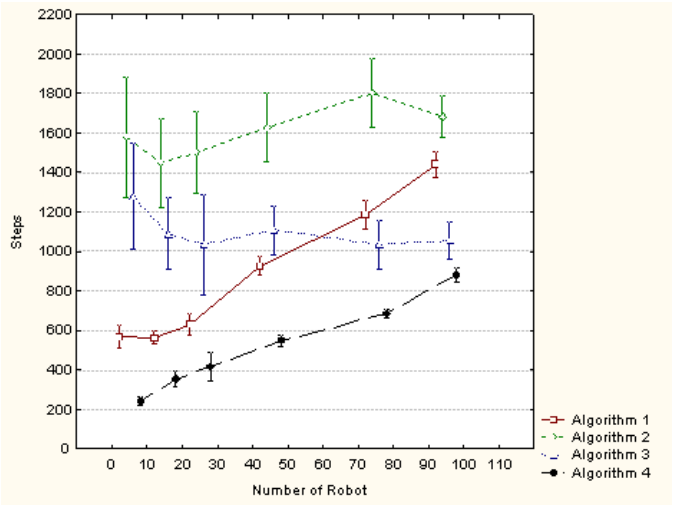

Fig. 3. Effect of robot number (Door number:1, Density:0.03, Door-Width:0.9m, Obstacle type:$1-1$)

In Figure 3, the condition was: Door number: 1, density: 0.03, door width: 0.9m, obstacle type: $1-1$. Data were analyzed using repeated measures ANOVA to compare 4 algorithms. There are significant differences among 4 approaches. For example, for algorithm 4 and 1, F(7,32)=142.0, p=0.000; for algorithm 1 and 3, F(7,32)=31.10, p=0.000; for algorithm 3 and 2, F(7,32)=26.41, p =0.000.

Of the four algorithms, algorithm 4, Move Along Longest Path, is the fastest at getting robots out of the room with algorithm 1, Wall Following, taking slightly longer. The Standard deviation for the two best algorithms is also smaller than the other two indicating more consistent performance. The higher standard deviation of algorithms 2 and 3 is likely due to their randomness in choosing move plans. When the number of robots is small, for instance 10, algorithms 4 and 1 show much better performance than algorithms 2 or 3. However, as the number of robots increase, the advantage decreases indicating that performance of algorithm 1 and 4 decrease as the robot number increases.

Figure 4 shows the effect of number of doors on the 4 algorithms. The results are intuitive in that as the door number increases, the time needed to evacuate decreases.

In Figure 5, we present the effect of density on performance. The results shows that the density impacts the performance trivially. As the density increases, no significant performance decrease is found.

In Figure 6, the effect of door width on performance is shown. It is clear that as the door width increases, the performance is improved for all algorithms except algorithm 1 which is less impacted by the parameter change. At the door width 0.4m, which means the door is just wider than the

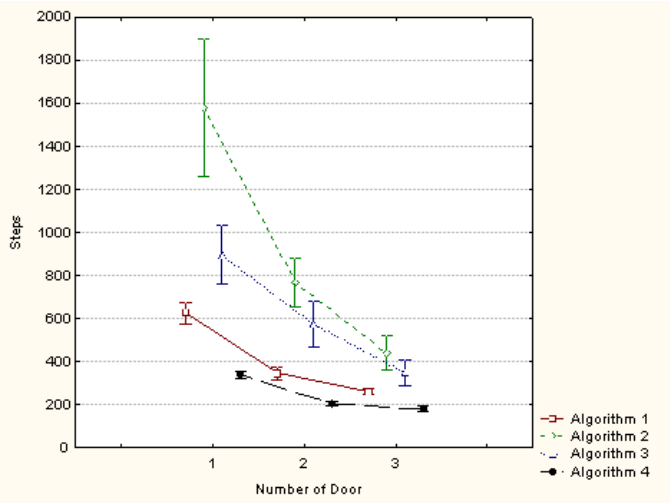

Fig. 4. Effect of door number (Robot number:20, Density:0.03, DoorWidth:0.9m, Obstacle type:$1-1$)

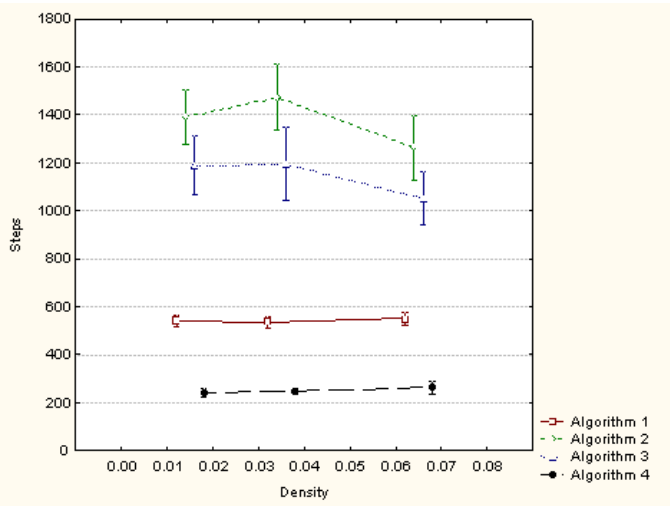

Fig. 5. Effect of density (Robot number:20, Door number:1, DoorWidth:0.9m, Obstacle type:$1-1$)

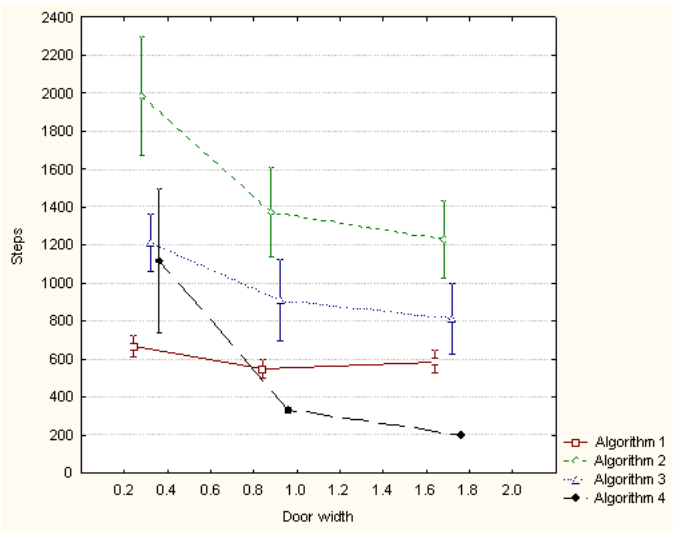

Fig. 6. Effect of door width (Robot number:20, Door number:1, Density:0.03, Obstacle type:$1-1$)

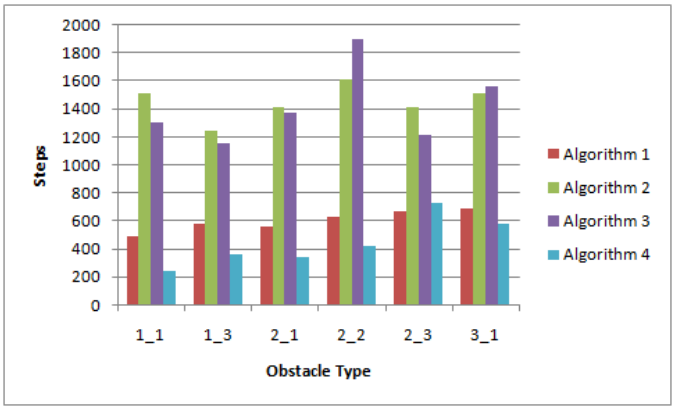

Fig. 7. Effect of obstacle (Robot number:20, Door number:1, Density:0.03, DoorWidth:0.9m)

robot, algorithm 1 takes the lead. As the door width increases, algorithm 4 performance improves making it the best of the algorithms.

In Figure 7, we show the effect of obstacle type to the performance. The obstacle number is from few (about 2) to many (about 15) and the obstacle size is from small (about the same size as the robot) to large (about 6 times the size of the robot). In most conditions, algorithm 4 takes the lead and algorithm 1 takes the second place.

IV. Discussion

Of the four strategies studied, Algorithm 4 in which the robot follows the longest path identified through its sensors achieves the highest efficiency. In our setting, the doorway provided the longest path that could be sensed, therefore, when Algorithm 4 robots reached the proximity of a door they were almost certain to escape leading to a higher efficiency than the other three methods in which robots could only exit if their prescribed movement led them through the door. However, it should be considered that this happens in the current theoretical simulation of environment, in which open space at the door way was assumed. Although the "longest path" heuristic was very effective in simulation it might meet less success in practice where a doorway might not always provide the longest laser scan path, i.e.; cut off by other walls or obstacles, e.g.; corridors. [3] reported a method similar to our Algorithm 1. Their study, however, was limited to the investigation of how a single robot could find its way out of a maze-like environment. For multi-robot systems, this approach has not been fully explored. In our study, by using this strategy, most robots in the simulation environment could make their way out of the room through the doors. As in the Figure 8 shows, after a brief period of time, most robots were moving along "walls", which might be obstacles, other robots or real walls. We also found that Algorithm 1 robots spent more time to find the doorway, which impacts their speed

of escape. This is because the robots have to take detours to find walls and then move along them. Although the speed of escape was slower, Algorithm 1 was largely unaffected by door width while Algorithm 4 was considerably slowed by narrow doors. Thus, Algorithm 1 may be more practical when a building is cluttered or damaged. This observation is borne out by the widespread use of wall following by firefighters and emergency responders under these same conditions. An additional advantage of Algorithm 1 is its minimal reliance on sensing. Because congestion problems of the sort we are studying are most likely to occur under harsh conditions, many forms of sensing and communication may be impaired. The ability of Algorithm 1 to achieve rapid diffusion of a robot team despite substantial impairment of sensing is a strong argument for its use. However, one limitation of Algorithm 1 is that when the robot approaches a wall-like obstacle near the center of the room it can get caught up in a cycle moving around it. The findWall() rescue routine increases the time for the robot to find the door, thus negatively affects the overall efficiency of the method. For the other two algorithms, 2 and 3, randomness plays a more significant role in the selection of movement. This randomness results in a surplus of to and fro movements of the robots; consequently they take more time to find the door and escape. Compared to Algorithms 1 and 4, these two random algorithms showed apparent disadvantages in our experiments and hence will not be recommended for use.

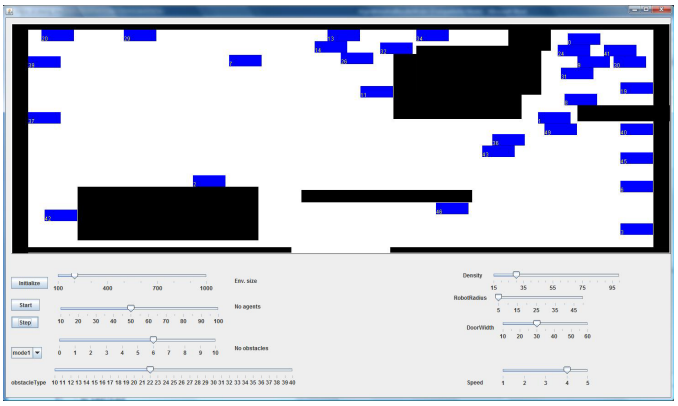

Fig. 8. Robots in Algorithm 1 are moving along the walls

V. Related Work

Path planning for multi robots in an unknown environment has been a challenging problem in mobile robotics. In[4], Kato et al suggested using traffic rules, such as “keep-right”, “stop at intersection”, “keep sufficient space to the robot in front you” etc to coordinate mobile robots. In [6], Harinarayan et al discussed motion planning of multi robots under the assumption each robot knows its current positions, able to distinguish a robot from an obstacle. In their reviews [5], Cao et al, gave a survey of some major theoretical issues such as, centralization/ decentralization , differentiation, communication structures etc, in cooperative robotics. In [7], Jaeger et al discussed deadlock detection and resolution for multiple mobile robots in the assumption each robot is able to communicate with others and know their current position. In addition, as an important field related to path planning, robotic mapping has been an active study area and gained increased research interest. However, “the dynamism of robot environments creates a big challenge”, and exploration in real time “is often solved sub-optimally via simple heuristics”, in [8] Thrun asserted in his survey of robotic mapping. As one of these simple heuristics, wall following algorithm was adopted in the maze solving problems in [3]. In [9], Braunstingl et al studied a wall following robot using a fuzzy logic controller so that the robot can move by the wall speedy and smoothly.

VI. Conclusions and Future Work

In future work, we plan to implement the wall following and longest open path algorithms on a team of 12 iCreate robots and evaluate the performance in a range of different environments. The algorithms will also be implemented in USARSim and used as a part of a human computer interaction testbed. We are also interested in looking at other possible algorithms, including algorithms utilizing communication. Finally, while the environment and problem is likely amenable to formal analysis, we intend to look at techniques that provide some bounds or expectations on performance, drawing on swarm modeling literature.

ACKNOWLEDGMENT

This research has been sponsored in part by AFOSR FA9550- 07-1-0039, AFOSR FA9550-08-1-0356 and ONR Grant N0001409-10680.

References

[1] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, Multi robot cooperation in the Martha project, *IEEE Robotics and Automation Magazine*, vol.5, Jan 1998.

[2] K. Konolidge, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset,D. Schulz, B. Stewart, and R. Vincent, Centibots: Very large scale distributed robotic teams, *In Procs. Int. Symp. on Experimental Robotics (ISER-04)*, 2004.

[3] C. S. Ananian and G. Humphreys, Theseus: A Maze-Solving Robot, *Independent Work Presented to the Department of Electrical Engineering at Princeton University*, May 23 1997.

[4] S. Kato, S. Nishiyama, and J.Takeno, Coordinating Mobile Robots by Applying Traffic Rules, *International conference on Intelligent Robots and Systems(IROS)*, pp.1535-1541, 1992.

[5] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng, Cooperative mobile robotics: Antecedents and directions, *Autonomous Robots*, 4:7-27 1997.

[6] K. R. Harinarayan and V. J. Lumelsky, Sensor-Based Motion Planning for Multiple Mobile Robots in an Uncertain Environment, *proc. of IROS'94, Munich, Germany* , pp.1485-1492, 1994.

[7] M. Jager and B. Nebel, Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots, *In Proc. of the IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS'01)*, 2001.

[8] S. Thrun, Robotic mapping: A survey, *In G. Lakemeyer and B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann*, 2002.

[9] R. Braunstingl, J. Mujika, and J. P. Ulribe, A wall following robot with a fuzzy logic controller optimized by a genetic algorithm, *Proceedings of the 1995 IEEE International Conference on Fuzzy Systems*, pp.77-82, March 1995.

[10] B. Yamauchi, Frontier-Based Exploration Using Multiple Robots, *Proceedings of the Second International Conference on Autonomous Agents*, pp. 47-53, Minneapolis, Minnesota, May 1998.