2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Planning for Multi-Robot Exploration With Multiple Objective Utility Functions

Jonathan Butzke Maxim Likhachev

Abstract—**With the continued improvements in portable computing power and sensing systems it is becoming more common for groups of robots to cooperate to achieve a goal. When the robots are operating in an initially unknown environment, the most natural form of cooperation is multi-robot exploration. For many years frontier based approaches have been commonly used to assign target points for each of the robots in the group based on expected information gain and distance to travel. In this paper we present an expansion to these approaches allowing for the incorporation of multiple objective utility functions that allow adjustment of the exploration priorities both for the individual robots and the group as a whole. In addition, we discuss real world results of our algorithm including our first place finish at the Old Ram Shed Challenge and second place at the MAGIC2010 main competition.**

I. Introduction

Before a group of robots can perform a task, they must first gain knowledge about their local environment. This may come in the form of an a priori model of the environment or it may have to be discovered in an online fashion. The exploration task requires that a collection of sensor robots systematically traverse an environment in order to locate static and dynamic obstacles. Trade-offs between the rate of exploration versus the thoroughness of exploration must be made in order to minimize the overall search time. In addition, the algorithm must be capable of working in any environment with a minimum set of assumptions on robot behavior and sensing capabilities. For single robots there are a variety of approaches that can be used to guide the robot towards unexplored areas. One of the most common approaches is using the concept of the boundary between the known areas of the map and the unexplored regions. The non-obstacle portions of this boundary are collectively known as the frontier [1] and are used as candidate goal points. Algorithms have been proposed that extend the basic frontier exploration scheme to multi-robot teams [2], [3], [4], [5] and outdoor environments [6].

When evaluating potential target points for the next goal location, most exploration algorithms use a combination of the information gain from sensing while traveling to the target and the cost of moving to that point. The information gain can be specified in several ways however, a typical method is to set the information gain for a state equal to a function of the knowledge about all states visible from that state. The information gain for a trajectory is equal to the sum of the information gain of all the states along the trajectory. It must be recognized that as each state is entered it may change the amount of certainty in surrounding states and thus lower the information gain for subsequent states along the trajectory. Due to this non-Markov property optimizing the information gain over trajectories is computationally expensive, and many algorithms estimate the information gain of a trajectory by using the information gain of the final point [4], [5], [7] while a few have attempted to explicitly include the expected information gain along the entire trajectory [3], [8]. Others have combined the information gain with additional features such as communications constraints [9] or improved physical models such as the likelihood of specular reflection from nearby obstacles [10]. When combined with other features the information gain is generically referred to as the utility of the state. Similarly, the cost can be viewed as a function of time, energy, or distance between the goal and the start state.

The combination of the states utility and its cost directly affect the order in which states are selected as potential goals. This selection process can be centralized or distributed and can be performed through the use of a market architecture [5] or through a high-level task allocation scheme using Petri Net Plans [11]. However the most straightforward method is a greedy assignment that selects the best robot-goal pair, assigns that to the robot, removes that robot from further consideration adjusting the utility of all states to reflect the assignment, and then repeats until there are no unassigned robots[12], [9]. This is the method we have adopted in our implementation.

The planning algorithm frequently has to take additional information into account, such as communication range limitations, sensor effectiveness and range, and terrain traversal costs. This paper introduces an algorithm for cooperative multi-robot exploration that utilizes a variant of a frontier based approach with the ability to specify specific hard and soft constraints on the robot through a central planning algorithm. We will specifically demonstrate a utility function incorporating soft constraints on maximum and minimum inter-robot distances and a heading bias. While a few approaches allow for a generalized multi-feature utility function

Affiliated with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 15217 and The GRASP Laboratory, University of Pennsylvania, Philadelphia, PA, USA, 19104. jbutzke@andrew.cmu.edu maxim@cs.cmu.edu. Funding in part by the Department of Defense through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

978-1-61284-456-5/11/$26.00 ©2011 IEEE

[13] ours differs in that we also allow for the creation of logical areas to influence the exploration priorities. We define these logical areas or regions as a set of states assigned to one or more robots (either specifically or ordinally, e.g. to the "first and second to arrive") and an associated weight matrix. This concept can allow a higher level planning system or expert input to influence the direction and areas of primary exploration. We will show an implementation that uses preferred and avoidance regions specified by human experts at intervals during run-time. Our implementation also calculates the information gain over the entire trajectory allowing for a more accurate estimate of a states utility. Our algorithm will be demonstrated on a two-dimensional grid based map, although it is expandable to any graph based representation of the environment. In addition, we show results of this algorithm applied to a team of eight robots in a combined indoor and outdoor environment as part of the MAGIC2010 challenge.

II. Algorithm

Overview

The algorithm attempts to find the "best" goal state in terms of the given cost function and then find the optimal route to that end state. The "best" state is dependent on distance/effort/time to arrive, amount of new information available at that state, and any adjustments based on the active constraints. To accomplish this, the algorithm used in our implementation can be divided into three primary stages: pre-processing of the input map, approximate ranking of the frontier states based on a heuristic, and forward simulation of the most promising goal states to determine the best assignment for each robot, as seen in Algorithm 1.

The pre-processing stage generates the coverage, cost, distance and information gain (IG) graphs for use in the later stages as shown in lines 2-4 of Algorithm 1. These graphs are used as the basis for conducting the search for a candidate trajectory. To increase the likelihood of choosing the optimal goal state each frontier state is placed in a sorted list ranked by the estimated utility score divided by the cost to reach that state as part of the second stage of the algorithm, lines 8-10. In the third stage, lines 11-22, the paths to these states are forward simulated in rank order and the best goal state is assigned to the corresponding robot. Since the IG is dependent on the trajectory taken, finding a globally optimal path is intractable for online real-time applications, instead we settle for its approximation.

The estimated utility score is composed of the information gain per unit traversal cost along with any bias terms related to the additional features and their associated weights. By using the information gain per unit traversal cost to select the next path, the algorithm may bypass isolated states while moving towards larger unexplored areas of the map. By applying a weighting factor to the traversal cost the ratio can be skewed to make information gain more or less important relative to the distance. In this way the thoroughness of exploration can be adjusted to meet the exploration requirements. The bias term has been introduced to control robot behavior such as minimum and maximum separation distances, and region assignments.

While our implementation uses a two-dimensional 8-connected grid map as our base graph, the algorithm is capable of operating on any graph based map.

Algorithm 1 Exploration Algorithm

```
1:  // pre-process maps
2:  [costMap,coverageMap] = process(map m)
3:  costMap = inflate(costMap)
4:  distMap = dijkstras(costMap, coverageMap)
5:  RR = remainingRobots()
6:  while RR.notEmpty do
7:     // calculate IG and rank states
8:     IGMap = infoGainEstimate(coverageMap)
9:     bias = calcBias(m)
10:    frontierHeap ← calcScore(RR, IGMap, distMap, bias)
11:    while t < planTime do
12:       // forward simulate each state
13:       pt ← frontierHeap.pop
14:       ξ = planPath(pt)
15:       score = evaluateTrajectory(ξ);
16:       if bestScore < score then
17:          bestScore ← score
18:          bestTraj ← ξ
19:          bestRobot ← pt.robot
20:          robotGoals[bestRobot] ← pt
21:       end if
22:    end while
23:    coverageMap ← coverageMap ∪ visible(ξ)
24:    RR.pop(bestRobot);
25: end while
26: return robotGoals
```

Stage 1: Map Processing

In the first stage of the planner, the input map is pre-processed to gather the necessary statistics for adequate ranking. The input map is an m by n grid with values proportional to the log probability of each state being occupied or free. The output is a set of array's.

The cost map is the first array and it represents the environment as an m by n grid with the value of each state proportional to the cost of traversing the corresponding state of the input map. This cost value can be augmented to include other factors such as time or energy to traverse the state. In our implementation the cost is proportional to the distance moved through the state. This cost map is augmented by a buffer applied to each non-traversable obstacle. This artificially increases the cost associated with traversing near obstacles; in effect forcing the robots to take longer paths but providing additional margin to collision. These parameters can be specified and updated at any time by the higher level systems to adjust the behavior of the robot.

The m by n by r distance map is the second array generated. The m and n dimensions are identical to those used for the cost map while the r is the number of robots

currently exploring. Each m by n layer, r_{id}, has values corresponding to the distance in meters from a given state to robot id as calculated using a standard Dijkstra's search algorithm on an 8-connected grid. For our implementation we did not allow traversal of unknown areas by setting the cost for all unknown states to $+\infty$ prior to running the Dijkstra search, however any non-negative weight can be given to unknown areas depending on the desired traversal characteristics.

The third and final array is the coverage map which provides an estimate of the IG possible from each cell. The IG is derived from the occupancy grid provided as input. As the absolute value of the information about a state grows higher (the state is more confidently free or more confidently an obstacle), the level of knowledge about that state increases. Each state k is assigned a value q_k that is inversely related to the level of knowledge of that state; in a state for which no doubt existed $q = 0$, whereas for a completely unknown state $q = 1$. As the robot learns more about a state k through sensing, the q_k value decreases towards zero. Since the value of q_k is a measure of the confidence of the sensor reading for state k, it can be converted directly from the log probability of occupancy (if using an occupancy grid) or derived from the sensor model. The IG then is calculated for a state j from the coverage map by equation 1.

$$ig_j = \sum_{k \in vis(j)} q_k \tag{1}$$

The function $vis(j)$ returns the set of states visible from a state x_j. For our implementation $vis(j)$ assumed a horizontally mounted laser scanner capable of sensing along rays emanating from the robot and extending out to the first obstacle taller than (and thus intersecting with) the plane of the sensor. This can also be a probabilistic model of the sensor suite, to account for less accurate sensors.

If a state k is visible from two states i and j, then the IG of i and j become dependent on the order the two are visited. Once the robot enters state i, it will gain knowledge of state k lowering its q value resulting in state j having a lower IG. Due to this interdependence between all of the available states, the IG for a given cell at a given time can only be estimated. Initially, we set $IG_{j,estimate}$ to the IG value ig_j assuming that the robot does not gain any information prior to reaching state x_j. This assumption leads to an over-estimation of the IG of a state.

Stage 2: Ranking Frontier States

The second stage of the algorithm is the determination of an ordered list of frontier states. A frontier state is a state that is known ($q = 0$) and that has at least one of its 4-connected neighbors with $q \neq 0$. Each frontier state, i is ranked based on the score calculated element-wise as in equation 2. The frontier list itself is simply a max-heap based on the score of each entry.

$$score[i] = \frac{(IG_{i,estimate})^{ExpTradeOff}}{(distance[i])^{(1-ExpTradeOff)}} \cdot bias_r \cdot bias_d \cdot bias_\gamma \tag{2}$$

The $IG_{estimate}$ and the $distance$ values are taken directly from the IG and distance maps, respectively, computed in the first stage. The $ExpTradeOff$ value is a user selected parameter that adjusts the relative importance of distance relative to IG. As $ExpTradeOff$ approaches 1 the algorithm prefers higher IG goals, as it approaches 0 it prefers lower cost (shorter) paths, and when $ExpTradeOff = 0.5$ it is neutral. The final term, $bias_{r \cdot d \cdot \gamma}$ is a robot specific term that is the product of three additional elements to control the goal selection process; region bias, distance bias, and repetition reward. Since this and the $distance$ term are robot specific, a given frontier cell will be in the frontier list as many times as there are robots exploring. Therefore, each individual entry on the frontier list specifies a robot-goal state pair.

During exploration with heterogeneous robots, it may be desirable to have a specific robot explore a specific subset of the environment. The algorithm handles this by allowing two distinct types of region assignment. The first is to assign a robot id to a given region, the other is to specify the number of robots allowed into a given region without specifying discrete id's.

The region bias term encodes this region information into the score of each frontier cell. In the first case, the region has a user specified set S of robots explicitly tasked to explore the area and a multiplier m specifying the level of attraction. For a robot numbered id the region bias is calculated by equation 3.

$$bias_r = \begin{cases} m & id \in S \\ \frac{1}{m} & \text{otherwise} \end{cases} \tag{3}$$

The second type of region has no specified set of robots assigned to explore it. For this type of region the bias is calculated by equation 4. This function allows for up to n robots to be in an area without penalty.

$$bias_r = \begin{cases} m & id \text{ is among the first } n\text{robot(s) in the region} \\ \frac{1}{m} & \text{otherwise} \end{cases} \tag{4}$$

Both types of regions reward robot-goal state pairs that place the robot either in their assigned regions or into a region without too many other robots as we have implemented it.

The distance bias only applies to robots in the same region. For these robots, a penalty is applied to any goal state evaluation that results in the robot being either too far or too close to another robot's assigned goal state. The distance bias is given by equation 5.

$$bias_d = \begin{cases} 1 & \text{if only robot in region} \\ 1 & D_{min} \leq d_{nearest} \leq D_{max} \\ \frac{d_{nearest}}{D_{min}} \cdot penalty & d_{nearest} < D_{min} \\ \frac{D_{max}}{d_{nearest}} \cdot penalty & d_{nearest} > D_{max} \end{cases} \quad (5)$$

D_{max} and D_{min} are the maximum and minimum ranges desired between a robot and its nearest neighbor. For our implementation, D_{max} is based on wireless coverage range, to prevent loss of network connectivity, while D_{min} is based on maximum sensor range, to minimize unnecessary overlap. The $penalty$ term is used to tune the relative importance of the distance bias term.

The final bias term provides $bias_\gamma$ to the robot for selecting the same goal as during the previous iteration. This term effectively requires a new goal state to be at least γ times better than the previous goal before the robot would rank it higher and thus change goal states. With this implementation adding penalties or rewards to the utility function based on additional features was made simple since all bias terms were multiplicative.

At the end of this stage of the algorithm, all of the frontier states are in a sorted list, once for each robot.

Stage 3: Forward Simulation and Goal Assignment

The third stage of the algorithm is to forward simulate the paths to potential goal states in order to determine a better estimate for the IG. During this stage the highest ranked frontier state-robot pair is selected from the frontier list and the trajectory is forward simulated to determine which states can be observed based on the current obstacle map. This IG is then used in equation 2 in place of $IG_{estimate}$ to determine the actual score for the state-robot pair. As processing time allows, additional frontier states are removed from the frontier list and forward simulated, with the best robot-goal state pair being saved. At the conclusion of the allowed processing time, the state with the highest score is paired with its associated robot, and that robot is removed from the list of eligible robots. The IG array is recalculated based on the predicted movements of the newly assigned robot and stage two and three are repeated for the remaining robots. This process repeats until the last robot receives an assignment. In this way each robot has received the best overall assignment that it could have from the states that were forward simulated.

Example

As an illustrative example, Figure 1 - 3 depict various snapshots of the exploration process. For this sequence, six robots were used with an effective sensor range of 7m. The $ExpTradeOff$ value was 0.5, multiplier m was 1000, $penalty$ was 1 and D_{max} and D_{min} were 50 m and 15 m, respectively. In Figure 1, the robots have just begun their exploration routines. The robots have separated into two groups with the spacing within each group at the lower end of the allowable window. By Figure 2 robot 5 has found one of

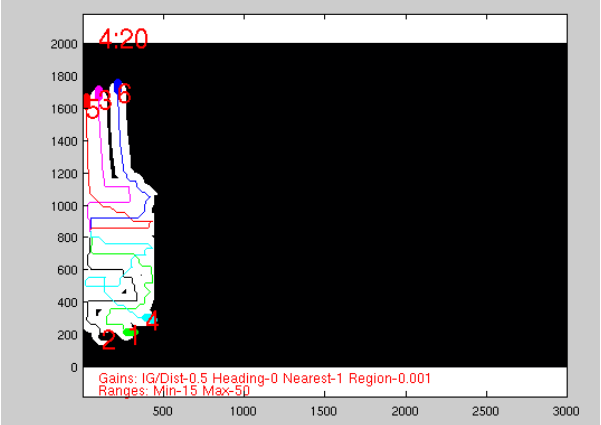

Figure 1. Six robots beginning an exploration simulation. The black area to the right is unknown, white is free space and the thick colored lines are the exploration trajectories for the next time step.

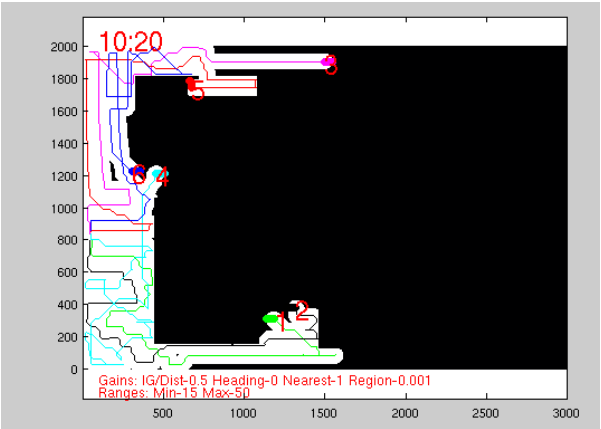

Figure 2. After ten minutes of exploration, robot 5 has entered a building area in the upper left corner.

the general exploration regions around an explorable building in the upper left corner of the image and has begun the exploration of the interior. After forty minutes of exploration, the map is largely complete with robot 3 surveying the other accessible building, and the other robots cleaning up islands of unknown regions as shown in Figure 3.

III. Results

A. Simulation Results

The algorithm was run on a series of randomly generated 2-dimensional grid maps similar to Figure 4 of varying sizes. During testing the minimum distance was set to the sensor range we saw in practice while the maximum sensor range was based on communications distances. A range of values was tried for all other user adjustable values and validated during subsequent real-world testing. The final values used for all of the tests were $ExpTradeOff$ set to 0.5, and $bias_r$ to 1000 and 0.001 for the first and subsequent robots, respectively. The distance bias penalty was set to 1 and $D_{min} = 20$ m and $D_{max} = 50$ m. The robot was given a sensing radius of 10 m and was allowed up to 0.3 sec of planning time during the forward simulation stage, resulting

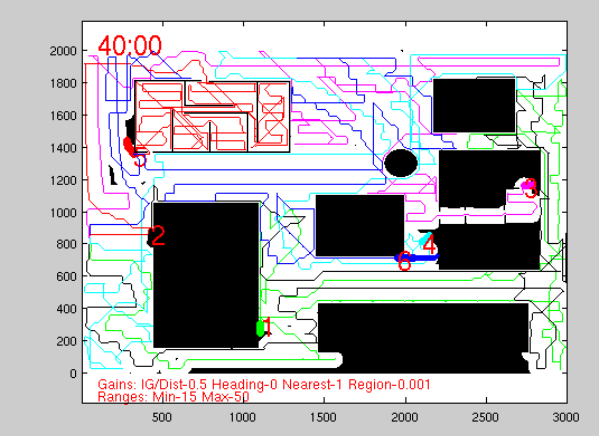

Figure 3. After 40 minutes of exploration. Robot 3 has now entered a building in the center right while robot 5 has finished the upper left building and resumed exploration outside. The medium thickness lines are exploration trajectories longer than one time step; the robot will complete the heavy line prior to the next trajectory evaluation.

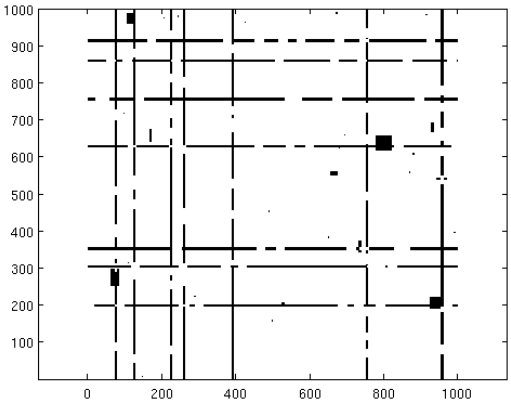

Figure 4. Example of the randomly generated map used to test the algorithm

in an overall system replan every 10 sec. The robots were also assumed to have a top speed of 1 m/s allowing 10 m maximum between replans. To test the algorithm, 5 robots were run on maps of 2000×2000 cells at 0.1 m per cell. The results can be seen in Figure 5 displayed as the fraction of the accessible space explored versus number of time steps. In addition, a combined indoor/outdoor map was generated to show the effect of logical area assignments and a hard inter-robot distance constraint, as shown in Figure 6 and 7.

B. Real Robot Results: MAGIC2010 Challenge

This algorithm was put to use on the University of Pennsylvania's team for the Multi-robot Autonomous Ground International Challenge 2010 (MAGIC2010) held in Adelaide, Australia in November 2010. For the main competition, a team of robots were required to enter an area comprised of indoor and outdoor environments, explore it, identify five different types of Objects of Interest (OOI's) and provide a detailed map with the objects locations pinpointed. In addition, for two types of objects, a specific sequence of

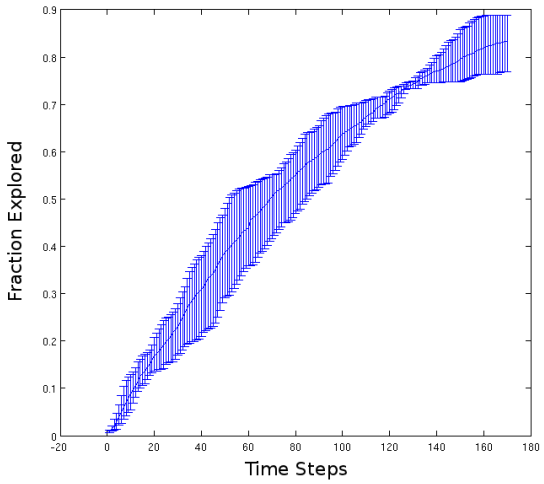

Figure 5. Exploration versus Time Step (10 second increments) for a 2000 by 2000 cell map at 0.1 m resolution and comprised of ~11% obstacle cells. Solid line represents the average fraction of free cells that have been detected while the error bars represent the maximum and minimum over the five random maps.

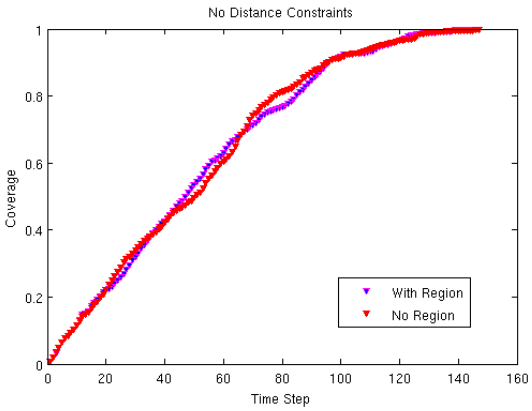

Figure 6. Comparison between coverage rate with and without regions for the case with no inter-robot distance constraints. When the robots are free to wander they perform about the same in both cases.

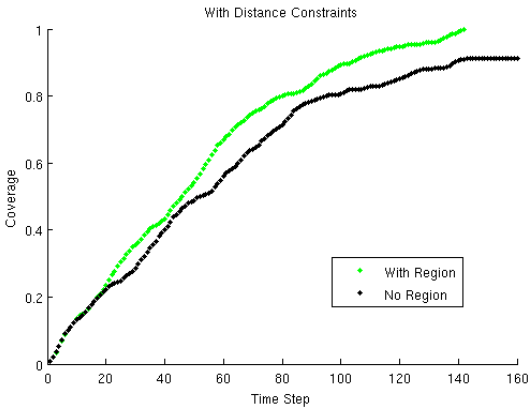

Figure 7. With distance constraints there is a clear advantage to specifying regions for exploration.

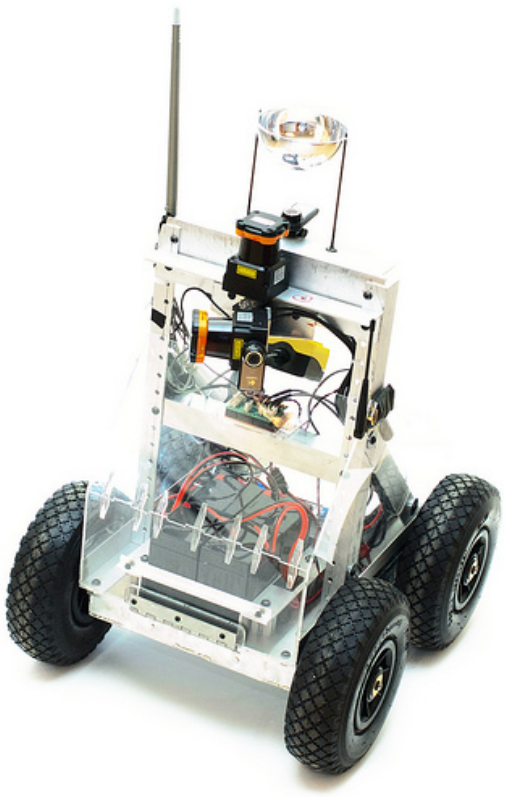

Figure 8. One of the University of Pennsylvania's MAGIC2010 robots. The robots used two laser range scanners, an omni-directional camera, and a panning camera to gather information about the environment. An on-board Mac Mini provided computational power, and a 802.11 antenna provided the data link with the ground control station (GCS). (Photo ©2010 Paul Vernaza)

actions had to be performed before a robot could move past the OOI. Two of the OOI types were mobile, while the rest were stationary. A second competition using a similar set of rules, but without any mobile OOI's and conducted entirely indoors was also held.

In the main competition, the team took 2^{nd} place. Unfortunately, unrelated mapping issues prevented the gathering of useful data regarding the exploration algorithm. However for the second competition, the exploration algorithm was able to completely map a space approximately 3600 square meters with 5 robots in under 35 minutes, identifying 9 of 12 OOI's in the process and garnering a first place for the team. For this implementation the algorithm was given five seconds total to process before having to provide an assignment for each of the robots. Computation occurred on a 2.80GHz quad-core i7 running Ubuntu 10.04. Eight threads were spawned to evaluate the frontier cells and perform other tasks concurrently. The playback of this event can be found in the accompanying video.

IV. Conclusions and Future Work

In this paper we have presented an extension of the frontier-based approach to multi-robot exploration that allows for the incorporation of multiple objective utility functions. This extension enables us to adjust the exploration priorities both for the individual robots and the group as a whole. The algorithm was implemented on a team of five physical robots that took the first place at the Old Ram Shed Challenge and second place at the MAGIC2010 main competition, both devoted to search and rescue operations.

Several of the steps of the algorithm can be performed concurrently, however, the requirement to forward simulate the paths is still the largest hold up in terms of being able to guarantee optimal results. While some of this is due to the inherent non-Markov properties of information gain, there remains some optimizations that could be done in order to process more possible frontier states during each planning cycle. Additional work to remove or at least diminish the central planning aspect remains to be done.

References

[1] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, Jul. 1997, pp. 146 –151.

[2] ——, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, ser. AGENTS '98. New York, NY, USA: ACM, 1998, pp. 47–53. [Online]. Available: http://doi.acm.org/10.1145/280765.280773

[3] R. G. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. L. S. Younes, "Coordination for multi-robot exploration and mapping," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, 2000, pp. 852–858. [Online]. Available: http://portal.acm.org/citation.cfm?id=647288.723404

[4] A. Visser and B. Slamet, "Balancing the information gain against the movement cost for multi-robot frontier exploration," in *European Robotics Symposium 2008*, ser. Springer Tracts in Advanced Robotics, H. Bruyninckx, L. Preucil, and M. Kulich, Eds. Springer Berlin / Heidelberg, 2008, vol. 44, pp. 43–52.

[5] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002.

[6] T. Tao, Y. Huang, F. Sun, and T. Wang, "Motion planning for slam based on frontier exploration," in *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, 2007, pp. 2120 –2125.

[7] H. H. Gonzalez-Banos and J.-C. Latombe, "Navigation strategies for exploring indoor environments," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 829–848, 2002. [Online]. Available: http://ijr.sagepub.com/content/21/10-11/829.abstract

[8] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 661 – 666.

[9] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 376 – 386, 2005.

[10] R. Grabowski, P. Khosla, and H. Choset, "Autonomous exploration via regions of interest," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, 2003, pp. 1691 – 1696 vol.2.

[11] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi, "Multi-objective exploration and search for autonomous rescue robots," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 763–777, 2007. [Online]. Available: http://dx.doi.org/10.1002/rob.20216

[12] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 2000.

[13] F. Amigoni and A. Gallo, "A multi-objective exploration strategy for mobile robots," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 3850 – 3855.