A Framework for Autonomous Mobile Robot Exploration and Map Learning through the use of Place-Centric Occupancy Grids

G. Michael Youngblood YOUNGBLD@CSE.UTA.EDU
Lawrence B. Holder HOLDER@CSE.UTA.EDU
Diane J. Cook COOK@CSE.UTA.EDU
Computer Science and Engineering Department, The University of Texas at Arlington, Arlington, TX 76019 USA

Abstract

The Autonomous Intelligent Knowledge-building Exploration (AIKE) system explores how an autonomous mobile robotic agent using sonar sensors and dead reckoning can learn a map of a completely unknown environment. AIKE generates grid-based maps based on the notion of a "place" (i.e., a location separated from other places by a confining gateway such as a door or entryway). Empirical trials show that the AIKE agent explores the environment with a faster rate of knowledge acquisition than random exploration, learning maps of higher accuracy by reducing error propagation. The algorithm produces a high level of similarity between maps of the same location created from different agent starting locations, making place recognition easier and providing consistency in mapping. We describe the AIKE system in the context of a proposed architecture for autonomous map learning, navigation, and map refinement. Our approach is best suited to operate effectively in large connected spaces such as office buildings using small mobile robots.

1. Introduction

In this paper, we explore an important aspect of a complete mobile robotic mapping system: autonomous exploration and initial map learning. Our goal is to learn a map of an unknown environment using a limited-sensor mobile robot with no *a priori* knowledge opening the way for more advanced techniques in continuous localization, knowledge usage, and map refinement to build upon this initial map knowledge.

Accurate representation of environments is important for robot deployment. Robots assigned a task involving travel within a specific environment will require either prior knowledge of the environment or a method of learning about the environment to allow successful navigation and accomplishment of the task. A robot must be able to maneuver within a designated area and interact with components in a dynamic manner in order to achieve advanced goals such as those involved in search and rescue missions, foreign environment exploration, mail/package delivery, guided tours, safety inspections, maintenance, and many other mobile tasks.

A fundamental aspect of creating advanced mobile robotic systems is the modeling framework for the robot's map. Intuitively, humans think about the notion of a place when they navigate. People are in one place, and often desire to go to another place, which they can usually visualize. The concern in this type of navigation is not the distance between two objects, or which path to follow, but identifying the next place needed to get closer to the desired place. A person may start in a room and want to go to another room. They can visualize where they want to go, and they can see where they are currently located. First, they find a gateway (i.e., a door or entryway) into another place. Upon gateway traversal, they are now in a new place. This procedure continues until the person reaches their destination place. The same idea can be applied to robotic navigation. If a robot perceives a room as a place and centers that place in its own occupancy grid, all other places can be mapped in a similar manner. Gateways can then be located on maps and connections between places can be topologically connected by the gateways, yielding a network of places that could represent buildings or other structured areas.

In our system, maps are learned using a combination of techniques. A primary search algorithm leads the agent in a rational manner around the space to create the place occupancy grid. A frontier (the boundary of an unexplored location) finding algorithm can assist if the primary algorithm leads the agent into a cove, and algorithms such as gateway determination, map smoothing, and centroid determination further assist this mapping.

We present an incremental approach to autonomous exploration and map building of an unknown environment by constructing occupancy grids (Elfes, 1989). We introduce a complete mapping, navigation, and updating system using a single or multiple robots, and describe a framework for place-centric occupancy grids connected in

a topological network called place-centric occupancy maps.

2. Conceptual Framework

We now describe our algorithms for map generation, use, and update encapsulated in the AIKE (Autonomous Intelligent Knowledge-building Exploration) and the PIKE (Parallel Intelligent Knowledge-building Exploration) systems. We use simplified occupancy as a basis for the maps. The notion of a place in the occupancy grids is introduced to make the occupancy grids place-centric.

2.1 Occupancy Grids

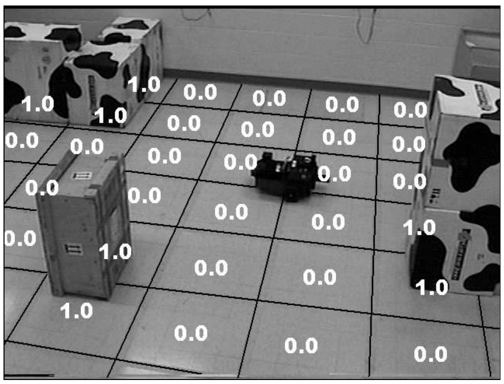

Figure 1. Occupancy grid overlay on an environment.

There are numerous papers describing in detail the specifics of the occupancy grid framework and creation dynamics (e.g., see Moravec, 1988). In our implementation, each cell within the occupancy grid is assigned one of three values: 0.0 (unoccupied), 0.5 (unknown), or 1.0 (occupied). Before mapping commences, a blank map is created consisting of a grid with all positions initialized to 0.5. The agent will then move around in the environment using sonars or other percepts to fill in grid evidence. Occupancy evidence is considered persistent. This prevents a majority of false readings from removing previous evidence of occupation due to having to rotate the robot in place for each pose due to < 360° sonar coverage around the robot. This technique may cause problems for noisy sonars to to placing irreversibly incorrect data in the map, so the sonar acquisition routines should get the best possible data before commiting the data. In our experiments we found this to be the best approach for our robots. Figure 1 illustrates a typical occupancy grid in the context of this work.

2.2 Place-Centric Occupancy Maps

Locations such as a specific room, hallway, lecture hall, etc. are important artifacts of a building or structure. We focus primarily on mapping structured areas such as these, and the inherent structure of such places consists of definite locations or places. The idea behind place-centric occupancy grids is to capture information about a single place in a single occupancy grid. This information is stored in a graph vertex. Places can be connected to each other through gateways, openings or connection points from one place to another place such as doorways, entryways, or portals. We create a topological graph by mapping place-centric graph vertices and connecting them through gateway edges. Figure 2 illustrates a building and the associated topological place-centric occupancy grid graph.

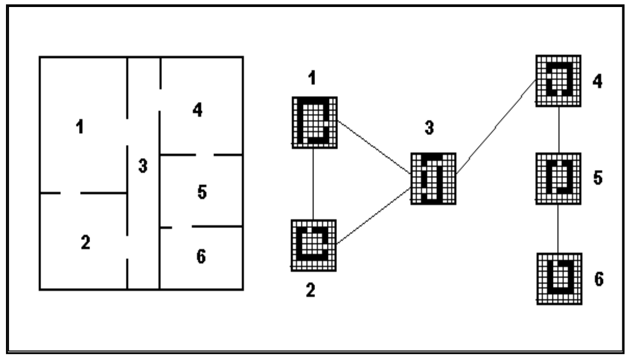

Figure 2. Place-centric occupancy grid.

There is only one place represented per occupancy grid. A map consists of many occupancy grids connected together. The use of a topological network for representation allows for easy use of path planning and search algorithms as well as other existing graph algorithms. As an extension, each vertex could store not only the place occupancy grid, but also information such as place features, names, and GPS coordinates. A series of maps could also be stored for tracking a dynamically changing environment.

2.3 The Autonomous Intelligent Knowledge-Building Exploration (AIKE) Agent

The AIKE agent architecture consists of three main modes of operation: exploration, navigation, and refinement. In exploration mode, the agent explores an environment and creates a map of that environment, the robot uses those generated maps in navigation mode to perform a task, and, since the world is dynamic, an agent in refinement mode will perform refinement of places to update or improve the maps.

Exploration involves creating the map of a building with no *a priori* knowledge. The first step is to open the map structure to allow for mapping of places. Then, the agent maps a place by creating a place-centric occupancy grid, determining what extra information needs to be collected for this place, and storing the map and all information in a structure. Next, the agent determines the gateways available from the place, stores this information in the structure, and then follows the gateways, mapping each location and creating structures to add to the map. Upon completion a machine-usable map representation is generated and stored for future use. Figure 3 illustrates this cycle.

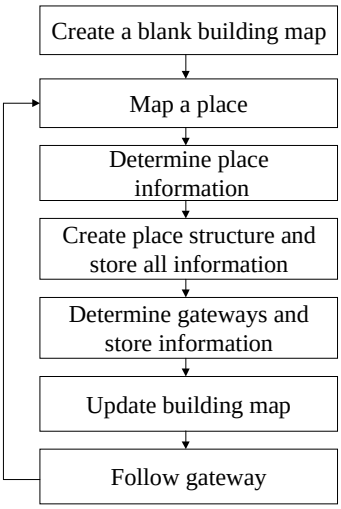

Figure 3. AIKE exploration mode flow.

Navigation mode is a utility mode. The agent is primarily interested in reaching the location required for the specified task. Once at the goal location, additional functions may be called on to perform tasks not defined by the AIKE software. Flags in the map structure can signal to the AIKE software that there may be unique handling requirements for the current place. For example, the agent may desire to use an elevator. An elevator handling routine may be called to allow the agent to use the elevator and then return the agent back to the AIKE software.

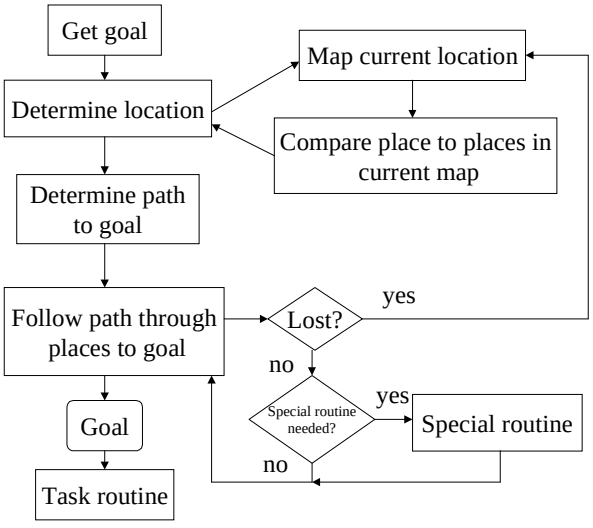

Figure 4. AIKE navigation mode flow.

Once a goal is given, the first step for the agent is to determine its location. It will perform exploration of the current place and then compare the current place to all of the places in the current map. The best match will be used as the starting location. A path to the goal place will be generated and traversed. Along the path, the agent will continually check its current location to make a best effort to ensure it is moving along the proper path. If the agent gets lost, it will initiate exploration and place matching to try to locate itself. If the agent cannot localize itself within a map it will resort to exploration to build a map. Once at the goal, the agent may execute additional outside routines to complete the task. Figure 4 illustrates the flow for navigation mode.

Refinement mode is a special case of the exploration mode. In refinement mode existing maps are updated to reflect changes and greater map detail. Continuous localization routines are used to ensure correct placement within the maps. The AIKE agent incorporates the necessary components for building, using, and updating robotic maps. It provides an extensible framework for mobile robot usage and task assignments combined with the key aspect of environmental mobility.

2.4 The Parallel Intelligent Knowledge-Building Exploration (PIKE) System

The PIKE System is an extension of the AIKE architecture for multiple agents. PIKE acts as a command and control center for multiple robot collaboration and coordination. It serves as a central repository for location maps, preferably stored by GPS coordinates. PIKE coordinates map refinement by granting requests for mode shifts by agents to refinement mode. In this manner only one agent may update a particular place and provide updated maps back to the PIKE System. PIKE monitors agents in exploration mode and can facilitate coordination efforts for potential map merging.

PIKE is implemented through a concurrent server that registers agents via Ethernet connections. A token system is used for each place within a map to allow for refinement mode shifts - only one agent is allowed to be in refinement mode per map place. The map library is maintained in a database and maps are issued to registered agents upon request. As newer maps are available, map updates are performed. Multiagent coordination consists of tracking each agent's position and preventing agents from colliding at gateways. PIKE can cause any registered agent to "pause" or "halt." PIKE also provides feedback to the observer on the status of all registered agents.

An advanced feature of the PIKE System is the ability to assist in parallel mapping of a building by performing map merging on generated maps. This can be accomplished by looking for structural similarities in maps and piecing a complete map together from parts. PIKE could direct agents to follow certain gateways that have not been explored, but identified by multiple agents. Throughout this paper, the "map" refers to a local map built by one agent. Multiple local maps can be merged to form a global map.

3. The Environment

Mapping in our experiments was conducted by two RWI Pioneer 1 robots. The robots are controlled by Pentium-based notebook computers running the Saphira 6.1f software with a C++ interface using the Pioneer Application Interface (PAI) libraries. All measurements were taken by instrument and were not estimated. For the current system, unknown environments are assumed to be comprised of orthogonal walls. For our experiments, borders are composed of either cinder block wall or reinforced cardboard. Gateways are classified as an accessible opening greater than 50 cm and less than 110 cm wide. Only one environment with a gateway was tested in our experiments, which does lead to another space.

Our algorithms were tested using a Pioneer 1 robot manufactured by Real World Interface, Inc. Due to a lack of 360° sonar coverage, in order for the Pioneer 1 robot to gain a complete view of its surroundings for a single position in space it must rotate in a complete circle for each pose. To compensate for sonar inaccuracies and specular reflections, we take a series of readings from each sonar and keep the lowest valid returned value. Each sonar reading updates occupancy values of grid locations that overlap the corresponding sonar cone.

4. Autonomous Mapping of Places

We now present the mapping mechanism used in the AIKE system. Algorithms for mapping, including the direction decision algorithm, as well as the assisting algorithms that find frontiers, smooth maps, and find gateways are described.

4.1 The Mapping Method

Maps are created by initializing an occupancy grid map for a certain place with occupancy values of 0.5. These values will be updated during exploration. Because sonar readings outside of two meters are often inaccurate, the map is updated by a 2 meter square local bounding grid. This local map, centered at the robot's current position (pose), slides around the map to perform updates. The size of each grid space within both the entire map and the local map is 10 cm². Figure 5 shows a sample occupany grid for a single robot with a map and moving local grid.

Mapping starts by collecting occupancy evidence from the robot's sonars and updating the local grid. Updating is performed by the interpretation of sonar readings as described in section 4. Due to specular reflections and noisy sensors, occupancy values are persistent. If occupancy was not persistent, overlap in sensor information would cause all local grids to appear falsely unoccupied. AIKE creates a grid map from occupancy evidence indicated by sonar readings. After the local grid is mapped, it is used to update the map. Using the map and the local grid the robot decides upon the next position to occupy, and moves to that location. The mapping continues until the place has been completely explored.

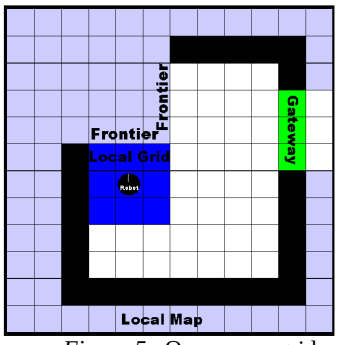

Figure 5. Occupancy grid.

Mapping completion is often hard to determine from information available within a system. The agent may be stuck in a cove and cannot see other unexplored areas. The boundary of unexplored territory is called a frontier (Yamauchi, 1998). If stuck, before the agent declares that mapping is complete, it should try to find any frontiers. This is accomplished by performing an increasing spiral search within the map starting from the current position of the agent. If there is a frontier, and thus more unexplored space, then the robot will need to move to new locations in the unexplored space. This means that a continuous localization and movement routine is necessary such as the method presented by Sebastian Thrun *et al.* (1998) or the Continuous Localization software available from the Navy Center for Applied research in Artificial Intelligence (www.aic.nrl.navy.mil/~adams/cont-loc; Schultz and Adams, 1998). For our experiments, mapping terminates when no frontiers remain in the agent's local grid.

After the map is complete and a place-centric occupancy grid has been created, some map cleanup may be needed. It is important that all generated maps look similar even if mapping started from distinct starting positions. All maps will need to be centered within the map by finding the centroid of the place and realigning it to the center of the map so that subsequent maps learned from the same place maintain the same reference point (their centroid).

At this point, gateways are determined, and mapping moves to the next place. Gateways are determined by identifying gaps in walls that meet a specified size threshold. New places connected by gateways are added to the search list. All of the mapped places should be linked into a topological map representing the current building or structure. A global tagging method (e.g., GPS) may be fixed to the network to assist in map retrieval based on global position.

The method presented in this study builds a map from a local grid moving with the agent through the current place until it is fully explored. Frontier finding, gateway determination, and centroid determination will improve map quality and assist in continuing exploration.

4.2 Primary Mapping Algorithm

Mapping is performed by functions *MapPlace* and *DetermineMoveDirection*. *MapPlace* first calculates the local grid position, then updates the local grid with sonar information and applies local grid information back to the map. Exploration is performed by determining the next direction for the agent to move. For our experiments, the robot moves 0.5 m at each step.

```
procedure MAP-PLACE( x, y, theta, localGrid, localMap)
  inputs: x, integer x location
          y, integer y location
          theta, integer (0-360) angle
          localGrid, 2D float array
[MAX_LOCAL_X][MAX_LOCAL_Y]
          localMap, 2D float array [MAX_X][MAX_Y]
  static: flag, boolean
          move, direction {N,E,S,W, neutral}

flag ← false
move ← neutral direction
        // Default valid direction -- non-biased to N,E,S,W

while move is valid direction
begin
  updateLocalGrid(x,y,localMap,localGrid)
  // Put local map info on local grid for location
  mapLocalGrid(localGrid)
  // Build local occupancy grid with robot
  updateLocalMap(x,y,localMap,localGrid)
  // Put local grid info on local map for location

  move ← determineMoveDirection(x,y,localGrid,localMap,move)
  // Determine next move

  if move is valid direction
    moveAgent(move,theta,x,y,localMap)
    // Move robot to new location
    flag ← false
  else if flag is true
    end
  else
    flag ← true
    move ← neutral direction
```

Figure 6. MapPlace algorithm.

The direction in which to move next is determined by the function *DetermineMoveDirection* (the choices are North, South, East, or West). The agent cannot move to an occupied space or back to the previous location. The linear distance to the closest large obstacle, unknown location or edge of the map is calculated for each of the four move directions. If an obstacle is closer than the other criteria, a terminal value is set indicating that mapping is complete in that direction. AIKE then favors the farthest distance of the remaining criteria, based on a desire to move to the farthest unknown. Whichever direction is available and has the most distant feature will be selected for the next move.

MapPlace moves the agent to the new location and continues mapping. If no valid moves are possible, it will map the same location. If two consecutive *DetermineMoveDirection* calls yield no valid moves, mapping is complete. Pseudocode for these functions is shown in Figures 6 and 7.

```
function determineMoveDirection(x,y,localGrid,localMap,move)
returns direction
inputs: x, integer x location
        y, integer y location
        localGrid, 2D float array
[MAX_LOCAL_X][MAX_LOCAL_Y]
        localMap, 2D float array [MAX_X][MAX_Y]
        move, direction {N,E,S,W, neutral}
 static: NorthRange, boolean; EastRange, boolean; SouthRange,
         boolean; WestRange, boolean; NorthDistance, integer;
         EastDistance, integer; SouthDistance, integer; WestDistance,
         integer; MaximumDistance, integer

for each direction in the order {North, East, South, West}
  {North, East, South, West}Range ← true if the block adjacent to
agent in that direction does not have any obstacles located in a box
bounded by a left and right value and a depth distance relative
to the agent within the localGrid; otherwise, false

for each direction in the order {North, East, South, West}
  {North, East, South, West}Distance ← the distance to the first
unknown grid location or end of the map in a box bounded by a left
and right value within the localMap

 **If the distance to an obstacle greater than a defined
threshold level of occupancy is determined before the other
criteria, then a terminal value indicating that mapping is
complete in that direction is assigned to that direction

MaximumDistance ← max{ NorthDistance, EastDistance,
                       SouthDistance, WestDistance}

if NorthDistance, EastDistance, SouthDistance, and WestDistance =
terminal values
        return NONE

if NorthRange = true and NorthDistance = MaximumDistance
        return NORTH

if EastRange = true and EastDistance = MaximumDistance
        return EAST

if SouthRange = true and SouthDistance = MaximumDistance
        return SOUTH

if WestRange = true and WestDistance = MaximumDistance
        return WEST

otherwise return NONE
```

Figure 7. DetermineMoveDirection function.

4.3 Gateway Determination

Finding the gateways within a map is important for linking connected regions. To identify gateways, we look for gaps within the map. Each sequence of unoccupied cells in a grid row or column that is greater than the wall smoothing threshold, but less than the maximum gateway size, is marked as a gateway. The procedure is then repeated for the columns of the map. This procedure will identify all unoccupied regions within a specified size that are vertically or horizontally aligned with the map as

gateways, and will add the area connected by the gateway to the exploration search list.

4.4 Wall Smoothing and Centroid Determination

These procedures are useful for accurate map generation. The wall smoothing algorithm fills gaps in the map marked as occupied space. The net effect is to fill holes due to sensor imperfections, and to smooth continuous walls and surfaces in the map. Wall smoothing is accomplished by marking grid cells as occupied when they are surrounded by occupied cells within a threshold distance away. The centroid function determines the center location for the bounding box of the mapped area. These values can be used to align multiple maps of the same place, regardless of the robot starting position.

The algorithms presented here create the foundation for the AIKE mapping capabilities. The next section presents the results of applying this approach.

5. Experimental Results

This section presents the results of seven different initial mapping trials we performed in experimentation, six used the algorithms presented in the previous sections and one replaced the direction decision with a random choice, tested using three different environments. These trials were used to map places to test the map learning ability of the base algorithms. Gateway determination and full map building system implementation is still in progress. All mapping exercises were performed in the UT Arlington Robotics Lab.

5.1 Error Propagation

As with many robots, the Pioneer 1 robots are subject to a certain amount of error in positioning and alignment. As the robot moves around the environment, these errors compound. Because we are exploring unknown environments, there is no frame of reference to adjust for these errors. Error propagation is reduced by careful calibration and ensuring that every trial starts with the robot facing true north, but error still exists in the mapped results due to these inaccuracies in robot movement.

5.2 Places

The experiments focus on mapping three created environments. The first is a simple rectangular room with a single gateway called the "Gateway Place." The second is a room shaped like the letter 'L' and is aptly named the " 'L' Place." The last is a room with irregular walls and an obstacle free-floating in the space. This last environment is called the "Complex Place."

All three environments have a baseline map created from the actual room layout, scaled to the mapped grid size. These maps are shown in Figure 8. All robot-produced maps were compared to these baseline images. The best fit was determined from the final map produced in each trial. Only pixels within the boundary of the room outline, inclusive of the boundary line, were compared. The number of exact matching pixels were counted at each stage and divided by the total number of baseline pixels to determine the level of accuracy.

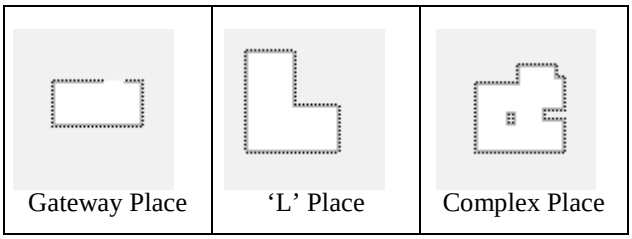

Figure 8. Baseline maps for tested places.

5.3 Gateway Place Experiments

The gateway place was used for three different trials. The first two trials used the algorithms presented in the previous section. The last trial replaced the *DetermineMoveDirection* choice of robot move direction with a random choice. Each trail had a different starting location in the place.

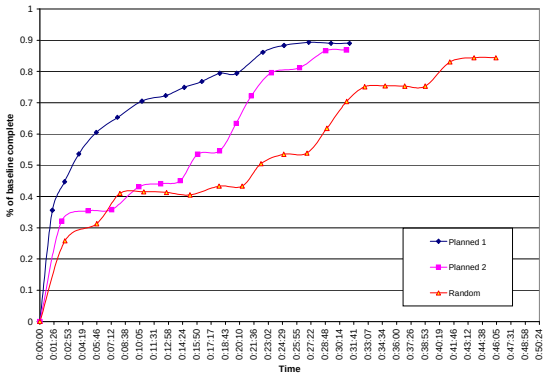

Figure 9. Gateway place map acquisition (AIKE vs. random).

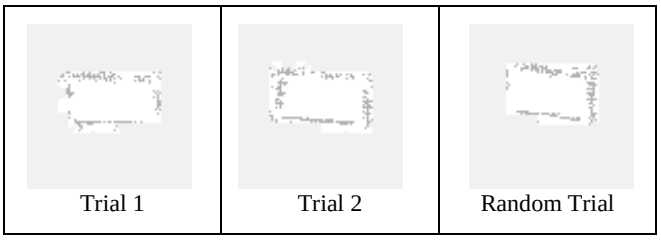

Figure 10. Final Mapping Results for Gateway Place

In this experiment, AIKE yielded an 89% accuracy to the baseline in 16 steps taking 31 minutes and 14 seconds for the first trial and 86.9% accuracy in 15 steps taking 32 minutes and 57 seconds for the second. Both of these results are better than the random exploration, which produced a map with 84.4% after 20 steps taking 46 minutes and 1 second. Figure 9 graphs the map acquisition rate for the three trials. Figure 10 shows the acquired maps. Trial one shows a 4.6% accuracy improvement, and completed mapping nine minutes and 11 seconds faster than the random trial. It is important to

map quickly to minimize error propagation caused by excessive robot movement.

5.4 'L' Place Experiments

The 'L' place was used for two different trials. Both trials used the AIKE algorithms presented in the previous chapter. Trial one took 35 steps yielding an 84.1% accuracy to the baseline in one hour 30 minutes and 28 seconds. Trial two yielded a map with 89.7% accuracy to a baseline in 38 steps taking one hour 41 minutes and 16 seconds.

A comparison of the mapping results for the 'L' place trials is presented in Figure 11. The rate of map acquisition is still high. Of interest are the plateaus in the acquisition plot for trial two. These are due to periods of time where the robot was backtracking over previously explored territory. During those times there was not a significant amount of knowledge discovery. Figure 12 shows the acquired maps.

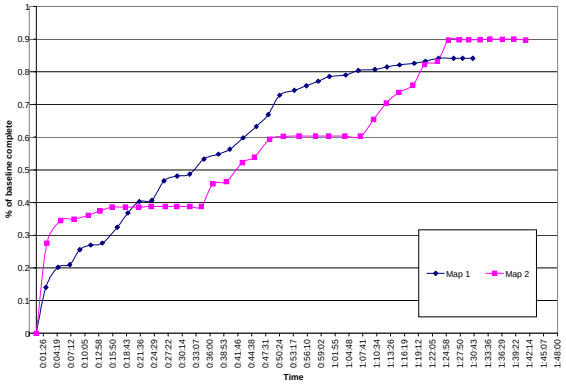

Figure 11. 'L' place map acquisition.

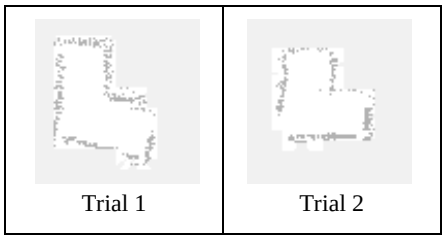

Figure 12. Final Mapping Results for 'L' Place

5.5 Complex Place

The complex place was used for two trials. Again, both trials used the AIKE algorithms. Trial one took 41 steps to finish with an 89.2% accuracy to the baseline taking one hour 37 minutes and 18 seconds. It took 35 steps to complete the map in trial two with a 90.0% accuracy taking one hour 21 minutes and 45 seconds. The composite knowledge discovery graph is presented in Figure 13.

These experimental results indicate that the AIKE algorithms yield more efficient explorations than random exploration, with good map acquisition accuracy. AIKE produces maps of high similarity even when starting from different initial mapping locations and using different robots. Although not experimentally presented here, the algorithmic framework we described for gateway determination produced many false gateways in experimentation, map smoothing proves to be only a marginal improvement when the initial data acquisition is good, but centroid centering greatly improved consistency between maps in the same referential frame. These algorithms represent a completely autonomous approach to exploration and map generation, without joystick control or predefined exploration paths.

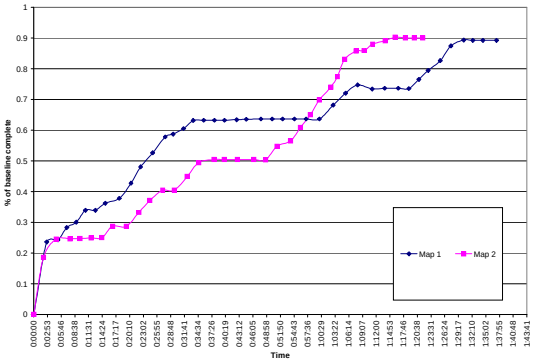

Figure 13. Complex place map acquisition.

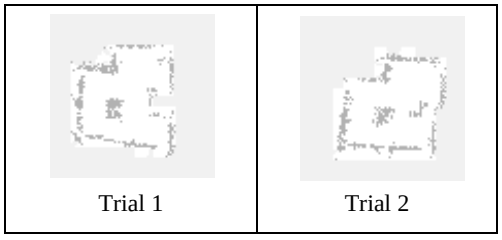

Figure 14. Final Mapping Results for Complex Place

A feature that bears note in the data is the increase in complexity of the mapping paths for robots in trials that had to explore in a predominately westward direction. Due to the directional bias in the algorithms, westward exploration is often hindered by the tendency to be pulled back to a northeasterly direction. Changing the directional bias or employing a heuristic could counteract this phenomenon. An adaptive directional bias approach may reduce mapping time and improve overall efficiency.

6. Related Work

Much research has focused on robot exploration and map learning. Many of the original work and subsequent discoveries in grid-based mapping have been done by Moravec (1988), Elfes (1989), and recently Yamauchi (1998). Kortenkamp (1994) performed some of the first experiments in gateway usage and determination. Yamauchi and Langley's (1996) work in place learning motivated our approach, but recently Sebastian Thrun (1998) has developed a system for continuous localization that gave us a need to fill. Our approach is designed to

provide the beginning maps for an advanced navigation and map refinement system so that a robot would not have to be "joysticked" through the environment, but could initially explore it on its own. PIKE was motivated by the multiple robot map learning collaboration in the research of Lopez *et al*. (1997).

7. Conclusions and Future Work

In this paper, we have demonstrated that a mobile robot can be used to build an accurate map with no *a priori* knowledge and no human intervention. In particular, we show that an algorithmic approach using a limited sensor mobile robot can produce maps that are 84% - 90% accurate to a known baseline of that environment. Two robots of the same type were used in experimentation—the 'L' Place trial 2 was run with a different robot than the other trials with similar results. The ability to produce maps of high similarity from different initial positions and with different robots is evident in the results. Autonomous mobile robotic exploration and mapping of place-centric occupancy grids is realizable as indicated by this research.

The initial results of this effort are promising, but also point to areas where continued research is needed. The immediate future work would involve the creation of a robust continuous localization system for the RWI Pioneer 1 robot, which will make possible implementation of the frontier finding algorithm and improved mapping capabilities. Accurate gateway determination is also necessary for the creation of the mapping structure described here and is a problem which when solved in combination with the algorithms and methods presented here will allow for the creation of a fully autonomous mapping system.

Acknowledgements

A note of gratitude to Dr. Pat Langley whose visit at UTA inspired the direction of this research.

References

Elfes, A. (1989). Using Occupancy Grids for Mobile Robot Perception and Navigation. *IEEE Computer*, June, 46-57.

Konolige, K. G. (1997). Saphira Manual Version 6.1.

Kortenkamp, D., and Weymouth, T. (1994). Topological mapping for mobile robots using a combination of sonar and vision sensing. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 979-984.

Kortenkamp, D., Bonasso, R., and Murphy, R., eds. (1998). Artificial Intelligence and Mobile Robots, Cambridge, MA: MIT Press.

Langley, P. and Pfleger, K. (1995). Case-base acquisition of place knowledge. *Proceedings of the Twelfth International Conference on Machine Learning*, 244-352. Lake Tahoe, CA: Morgan Kaufmann.

Langley, P., Pfleger, K., and Sahami, M. (1997). Lazy Acquisition of Place Knowledge. *Artificial Intelligence Review*, 11, 315-342.

Lopez, M., Lopez de Mantaras, R., and Sierra, C. (1997). Incremental map generation by low cost robots based on possibility/necessity grids. *Proceedings of the Thirteenth International Conference on Uncertainty in AI*, 351-357.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environmental mapping. *Autonomous Robots*, 4, 333-349.

Moravec, H. P. (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 61-74.

Moravec, H. and Blackwell, M. (1993). Learning Sensor Models for Evidence Grids. *CMU Robotics Institute 1991 Annual Research Review.*

Schultz and Adams. (1988). *Continuous Localization Using Evidence Grids*. Proc. of the IEEE International Conference on Robotics and Automation, IEEE.

Thrun, S., Fox, D., and Burgard, W. (1998). A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Machine Learning*, 31, 29-53 and *Autonomous Robots*, 5, 253-271 (joint issue).

Thrun, S., Gutmann, J., Fox, D., Burgard, W., and Kuipers, B. (1998). Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 989-995.

Yamauchi, B. (1998). Frontier Based Exploration Using Multiple Robots. *Agents*, 47-53.

Yamauchi, B. and Langley, P. (1996). Place Learning in Dynamic Real-World Environments. *Proceedings of RoboLearn-96: International Workshop for Learning in Autonomous Robots*, 123-129.

Youngblood, G. M. (1999). Autonomous Mobile Robot Exploration and Mapping of Place-Centric Occupancy Grids, thesis. The University of Texas at Arlington.