

## A cognitive agent for searching indoor environments using a mobile robot

Scott D. Hanford

Lyle N. Long

The Pennsylvania State University  
Department of Aerospace Engineering  
229 Hammond Building  
University Park, PA, 16802  
814-865-1172  
sdh187@psu.edu, lnl@psu.edu

Keywords:

Cognitive architectures, Robotics

**ABSTRACT:** *Recently there has been increased interest in the use of cognitive architectures for mobile robots. In this research, a cognitive agent (developed using Soar) has been developed to perform an indoor search and rescue mission using a mobile robot. For this application, sensor processing systems such as image processing, fuzzy logic, and image matching are used to generate information about the environment. The cognitive agent described in this paper is capable of using this information about the environment to explore a building while detecting and recording the locations of common types of intersections, making decisions about where to go based on these intersections, and detecting objects of interest and recording their locations. Results are given for a test in a simple environment.*

### 1. Introduction

An important problem for mobile robotics is conducting a search within an unmapped building for specific objects. The RoboCupRescue search and rescue competition ([www.robocup.org](http://www.robocup.org)) has provided a showcase for this application. For search and rescue in an unknown building, the ability to create a map of the environment, ideally with symbolic information that could easily be shared with humans such as first responders and the ability to identify objects of interest and report where they are located within the environment have been identified as important capabilities (Balaguer, Balakirsky, Carpin, & Visser, 2009).

Most mobile robots and unmanned vehicles are currently either remotely operated or designed for specific tasks. Cognitive architectures describe the general structures and processes that are thought to be needed for intelligent or autonomous behavior. These architectures can be used for a variety of tasks, using specific knowledge encoded for each task. Additionally, since these architectures aim to create systems that think like a person, it has been argued that cognitive architectures will be useful for collaborating and sharing information with humans (J. E. Laird, 2009; J. Gregory Trafton et al., 2006).

The use of cognitive architectures for robots was documented as early as 1990 (J.E. Laird, Hucka, Yager, & Tuck, 1991; J.E. Laird & Rosenbloom, 1990), but has recently become more popular (J. E. Laird, 2009). Researchers have used both popular cognitive architectures such as ACT-R (Anderson et

al., 2004) and Soar (J.E. Laird, Newell, & Rosenbloom, 1987) and developed their own architectures to use on robots. ACT-R and its various extensions have been used to study human-robot collaboration (J. Gregory Trafton et al., 2006). Soar has been used, along with robot schemas and a natural language system, to develop ADAPT (Benjamin, Lyons, & Lonsdale, 2004). SS-RICS is a robotic control system that uses several AI techniques and a production system similar to ACT-R (Kelley, Avery, Long, & Dimperio, 2009). In addition to these physical robot applications, cognitive architectures have also been used to control simulated unmanned vehicles. For example, Soar has been used to autonomously fly U.S. military fixed-wing aircraft during missions in a simulated environment for the TacAir-Soar project (Jones et al., 1999).

There has also been progress in the development of robotic maps with symbolic information, inspired by the idea of cognitive maps (Beeson, Modayil, & Kuipers, 2010; Tomatis, 2008). Research in the area of cognitive maps has generally agreed that topological information including landmarks and paths between these landmarks are important to human navigation (Chown, Kaplan, & Kortenkamp, D., 1995; Kuipers, Tecuci, & Stankiewicz, 2003). Information about meaningful landmarks (e.g., intersections) has also been of interest in robotics because they can be used to complement the metric representations typically used in robotic maps and have the potential to reduce errors commonly present in metric robot maps (e.g., from odometry drift). However, the generation of useful topological maps has faced the symbol grounding problem: being able to reliably abstract "useful

symbols from continuous, noisy perceptions of the environment, i.e. how to reliably detect and recognize places and paths (Beeson et al., 2010).” For example, earlier approaches have included designating a landmark every time a robot has traveled a specific distance without regard to the features at that location or when a human operator instructs the robot that a landmark is present. More recent and useful approaches have used features extracted from sensor information (Tomatis, 2008) or probabilistic metric representations of the local environment (Beeson et al., 2010) to generate meaningful landmarks autonomously.

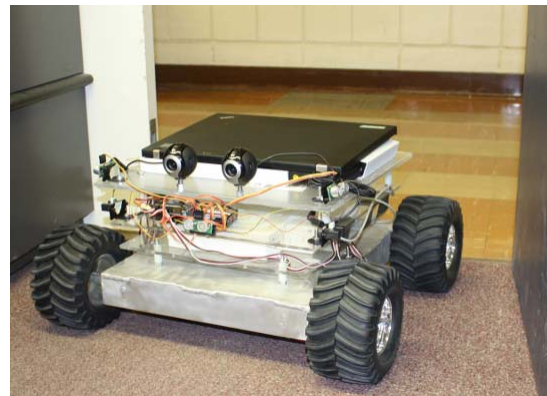
Two of the robotic systems utilizing cognitive architectures mentioned above have described systems that include cognitive maps or topological information. Kennedy et al. described a robotic system that used ACT-R for a collaborative reconnaissance mission (Kennedy et al., 2007). This system uses multiple layers of maps including occupancy grids that were used primarily for navigation and obstacle avoidance and a cognitive map that represents space in a qualitative manner as a 2-D grid in which objects are placed using the metric information describing their locations. The 2-D cognitive map supports symbolic reasoning about relative locations of objects of interest (such as teammates, targets, and buildings). Kelley et al. described how data from a laser rangefinder was used to identify common intersections in mazes within SS-RICS (Kelley et al., 2009). Information about the detected intersections was then used to make navigation decisions while using a maze searching algorithm.

Combining the use of a cognitive architecture with the ability to generate a map with symbolic information could result in a system with the potential to be quite useful for search and rescue missions. This paper describes a robotic system that uses a Soar cognitive agent to perform search inside a building, while generating symbolic information about the building's structure. While different cognitive architectures focus on different aspects of cognition, there are several strengths of Soar that make it an excellent choice for the system described in this paper. The Soar cognitive architecture has an automatic mechanism for creating subgoals that facilitates both the hierarchical organization of operators and automatic planning when a Soar agent is unsure about the best action to choose. The Soar architecture also defines a decision procedure that separates operator proposal selection and application. This feature supports general operator proposals that do not need to consider every possible situation while allowing proposed operators to be evaluated and compared to other proposed operators for specific situations. Soar is capable of scaling to the

large amount of knowledge needed for real-world problems, a characteristic which has not been shown by other architectures with a focus on detailed cognitive modeling (J. E. Laird, 2009). This characteristic will be useful for adding more sensors and more rules to the system described in this paper.

## 2. The Cognitive Robotic System

The Cognitive Robotic System (CRS) has been developed to use the Soar cognitive architecture on two mobile robots: a six-legged robot and a wheeled robot (Hanford, Janrathitikarn, & Long, 2009; Long, Hanford, & Janrathitikarn, 2010). The wheeled robot, called the SuperDroid and shown in Figure 1, was used for the work described in this paper. The hardware used for this research includes a laptop, microcontrollers for sensor integration and motor control, wheel encoders, sonar and infrared distance sensors, and a camera (more details about the specific hardware are available in (Hanford et al., 2009; Hanford & Long, 2010)). This section will describe how information from this hardware has been used to generate information about the environment that can be used by a Soar agent for a search and rescue mission.



**Figure 1:** A picture of the SuperDroid wheeled robot, in an office doorway.

The system used here has three sonar distance sensors (facing forward, to the left, and to the right) and two infrared distance sensors (facing to the left and right) on the robot. These sensors measure the distance to the closest obstacle in their perceptual field. The sonar sensors have a wide perceptual field and are useful for sensing space without obstacles while the infrared sensors have a narrow perceptual field and are useful for sensing small openings (e.g., a gap in a wall). Results from each of the five distance sensors are sent to the Soar agent's input link.

Encoders installed on the front wheels of the SuperDroid are used to estimate the robot's state: its location and orientation. The state estimation is

susceptible to drift affecting the accuracy of the estimation. The estimated x and y position and orientation of the robot are used by the Soar agent.

The estimated robot state and measurements from the distance sensors are used to generate an occupancy grid (described in detail in (Hanford & Long, 2010)). The occupancy grid generated in the CRS is a local grid (3m by 3m) that is centered on and moves with the robot. The grid provides a framework to fuse information about obstacles from all five distance sensors. A virtual rangefinder (similar to (Mozos & Burgard, 2006)) was developed that uses the occupancy grid as a virtual sensor. The occupancy grid is divided into 120 arc regions (each region is three degrees wide) and the virtual rangefinder is placed near the center of the occupancy grid. The obstacle in each region that is closest to the location of the virtual rangefinder is identified. The 120 arc regions are separated into 12 groups (of 10 regions each) and the distances within each group are averaged. The 12 average distances are sent to the Soar agent.

The Hough transform (Duda & Hart, 1972) is a popular technique that can be used to identify the dominant lines in an image. In the CRS, Hough transforms are used to identify lines that are likely to correspond to walls in the occupancy grid. The local occupancy grid is divided into two images: one that contains the occupancy grid to the left of the robot and a second that contains the grid to the robot's right. Hough transforms are used to identify the dominant line in the occupancy grid and whether that line is to the left or right of the robot. Then the most dominant line on the other side of the robot that has a similar orientation as the most dominant is identified. The parameters that describe these two lines (their orientation and perpendicular distance to the origin of the occupancy grid image) and a measure of how prevalent a line is in the image are sent to Soar from the CRS. If each line is prevalent enough (above some threshold), the Soar agent assumes the robot can use the orientations and distances to estimate the width and orientation of the hall in which the robot is located.

In addition to being used by the Soar agent, the results from the virtual rangefinder are used by a fuzzy logic system to identify common types of intersections (for example, T intersections of three different orientations 'T', 'Tr', 'Tl'; right turn 'R'; left turn 'L'; dead end 'D', and straight halls 'S'). The fuzzy logic system uses the three (of the 12) average ranges corresponding to the regions of the occupancy grid in front of, to the left of, and to the right of the robot to calculate the confidence that each of the seven above mentioned intersection types are present. The fuzzy logic system uses information from the Hough transform to estimate the

width of the current hall (or 'S' intersection), allowing the system to adapt to different width halls and intersections. The Soar agent receives the current confidence that each of the intersections is currently present.

The Scale Invariant Feature Transform (SIFT) (Lowe, 2004) is a computer vision algorithm that can be used for image matching. SIFT detects local image features and creates descriptions of the region around these features that can be used to match the features between different views of the object or scene containing the features, even in the presence of scaling, rotation, occlusion, and small variations in illumination and 3D camera viewpoint" (Lowe, 2004). The CRS has an image of the object (or images of the objects) to be searched for and looks for SIFT matches in images taken using the webcam on the robot. The number of SIFT matches found for the object(s) of interest are sent to the Soar agent's input link.

### 3. Soar Agent

#### 3.1 Agent Overview

The Soar agent described in this paper is capable of moving through a building, recording intersections and making decisions about where to go based on these intersections, and detecting objects of interest and recording their locations. Figure 2 shows the higher levels of the operator hierarchy for this agent. The initialize operator sets up some of the structure in working memory and initializes the Soar agent with information about its search task. The search operator is proposed until all of the desired objects are found. The record-intersection and record-object operators are proposed when intersections or objects of interest are detected. These record operators are preferred over the search operator and will be selected before the search operator if both operators are proposed.

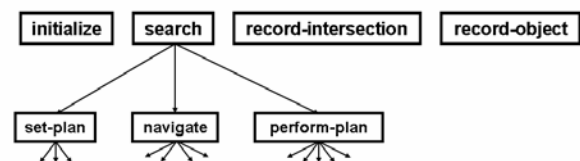


Figure 2: High levels of operator hierarchy in the Soar agent.

In the future, a return-home operator in which the agent will be able to use the recorded intersections to return to the area where the indoor search started will be added to the top level of the operator hierarchy. The operators for recording intersections and objects will also be used when future top-level operators besides search (e.g., return-home) are active.

There are a number of things the Soar agent remembers about its environment and its recent past. The agent records information about the intersections and objects it detects (when and where the detection occurred, how many intersections or objects have been previously detected) that will be useful for creating a topological map. All plans the agent creates are remembered. The reason the plan was created, the goal of the plan, the component actions needed to complete the plan, and information about the status of the plan (the time it was started, if the plan was successful, etc.) are also recorded. Information about the agent's recent environment is remembered and used to make decisions based on the current situation and recent history. For example, the agent remembers any intersections that were detected in the last five time steps so it can differentiate between a single, potentially incorrect, intersection detection and repeatedly detecting the same type of intersection.

### 3.2 Search

The search operator has three operators that are used to explore a building using two types of navigation strategies. The navigate operator uses reactive navigation strategies such as following a wall while the perform-plan and set-plan operators support navigation between waypoints.

#### 3.2.1 Navigate

The navigate operator is proposed if the agent detects it is in a hall ('S' intersection) or if no intersection is detected. This operator uses four suboperators (follow middle, follow wall, follow orientation, and avoid obstacle) to keep the robot in a good position to detect intersections as it moves through a building. Because of the limited sensors available on the robot, intersections are most easily detected if the robot stays near the middle of hallways with an orientation aligned with the hall.

The proposals of the suboperators in navigate are dependent on the presence or absence of walls to follow on the left and right sides of the robot. The Soar agent considers three cases that might represent the absence of a wall to follow: an infrared sensor returning a maximum measurement, a maximum virtual rangefinder measurement for the region to the side of the robot, and a side sonar sensor returning a measurement that is large compared to the current hall width (as determined by Hough line information). If any of these three scenarios are true, the agent assumes that there is not a wall to follow. If there are walls to follow on both sides of the robot, the follow middle operator is proposed. If there is a wall to follow on only one side of the robot, the follow wall operator is

proposed. If there is not a wall on either side of the robot, the follow orientation operator is proposed.

The above three operators assume that there are no obstacles in front of the robot. If the front sonar sensor detects an obstacle, the avoid obstacle operator is proposed. Currently, the avoid obstacle has the agent wait until the presence of the obstacle is reflected in the robot's occupancy grid. Once the obstacle is represented in the occupancy, a new intersection will be detected and a plan will be created as described in the next section.

#### 3.2.2 Setting plans in intersections

The set-plan operator is proposed when a new intersection is recorded. The goal of this operator is to create a navigation plan that will move the robot through the intersection. To achieve this goal, the Soar agent uses knowledge about what plans are useful in specific intersection types. All relevant plans are proposed for the current intersection type (e.g., for an 'R' intersection, turning to the right and turning around are proposed). One plan is selected based on each operator's preferences. Currently, a plan is selected at random (except for the plan of turning around, which is selected only if there are no other options), but in the future the operator preferences could be augmented by a search strategy such as preferring to go to an area that has not been visited before.

In the current Soar agent, these navigation plans consist of one or more waypoint goals, each of which is defined by a desired robot state. For navigation plans for turning (to the left, right, or completely around) in intersections except for dead ends ('D') there are three goals. The first waypoint goal is set to try to get the robot in the center of the intersection. This goal puts the robot in the center of the hall the robot is supposed to turn into and also allows the agent to confirm that the intersection that was recorded to trigger the navigation plan is correct (a 'R' intersection can sometimes be mistaken for a 'TI' intersection if the front wall is too far away or a 'R' intersection can be mistaken for a 'D' intersection if the front wall is sensed before the opening to the right). The second goal in a turn right or turn left plan is to turn to the desired orientation and the third goal is to travel straight forward following the robot's new orientation to get out of the old intersection. Other plans have less than three goals. Plans for turning around in a dead end consist of two goals: moving forward to confirm the intersection is a dead end and then turning around. Plans for going straight through an intersection only have one goal: moving forward.

The set-plan operator also is able to figure out what to do if the agent is in the middle of a plan and senses a new intersection of a different type than the intersection the plan is intended to move through.

### 3.2.3 Performing plans using waypoint navigation

Once a plan has been generated, the perform-plan operator attempts to execute the plan. There are operators to update the status of plans and goals, to change the goal the agent is attempting to accomplish, to calculate the waypoint state for the current goal, and to move the robot to a desired waypoint. Desired waypoint states are calculated using available knowledge about the environment, including the width and orientation of the previous hall and the distance between the robot and the wall in front of it (if a wall is present), and the robot state.

Once a goal waypoint is calculated, the agent moves the robot to the waypoint using operators that align the robot with the desired orientation and move the robot closer to the waypoint. When the robot is at the desired waypoint (as defined by thresholds for the distance and orientation to the waypoint), the goal is considered to have been achieved. At this point, the agent updates the plan and goal statuses.

### 3.3 Recording intersections

The record-intersection operator is used to record the type of intersection the robot is in and other information such as the time and location the intersection was first detected. Because of imperfect knowledge of the environment (sensing limitations), limits on when intersections can be recorded are used by the Soar agent to limit incorrect detections of intersections. For example, the accuracy of the occupancy grid is best when the robot has been moving in the same direction for at least a small distance (greater than 0.25 m). When the robot is turning or has recently turned a large amount, the occupancy grid may not be a good enough representation of the environment around the robot to allow accurate intersection detection. Because of this limitation, the Soar agent does not record intersection detections during or immediately after turning a large amount to reach a waypoint goal. Additionally, the Soar agent requires the robot to be in a position in which its sensors can accurately represent intersections on an occupancy grid (e.g., aligned with a hallway, near the middle of a hallway) before recording intersections.

Once the intersection has been recorded, an operator in the lower levels of the agent's operator hierarchy is used to update the time and location of when the intersection was last detected. The information

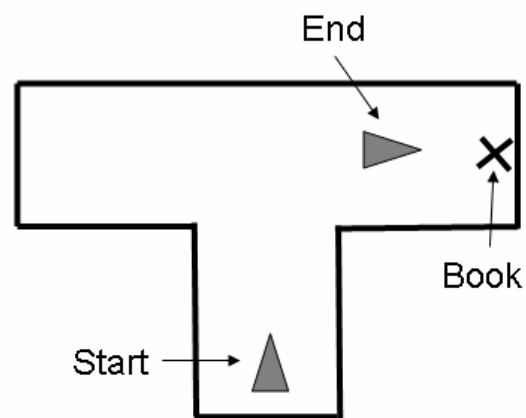
recorded about the intersection can be used later to create or update topological or cognitive representations of the building.

### 3.4 Recording objects

The record-object is proposed when the number of SIFT matches between the current camera image and an image of an object of interest exceeds a threshold. The type of object and the time and location when the object is detected are recorded in working memory. As described in the recording intersections section, once the object has been recorded, an operator in the lower levels of the agent's operator hierarchy is used to update the time and location of when the object was last detected. The information recorded about the object can be used to describe the object's location relative to the intersections that have been recorded.

## 4. Results

This section describes the results from a test in a simple T-shaped environment (shown in Figure 3) in which the Soar agent described in Section 3 controlled the SuperDroid robot described in Section 2. The object of interest the agent searched for during this test was a specific book.

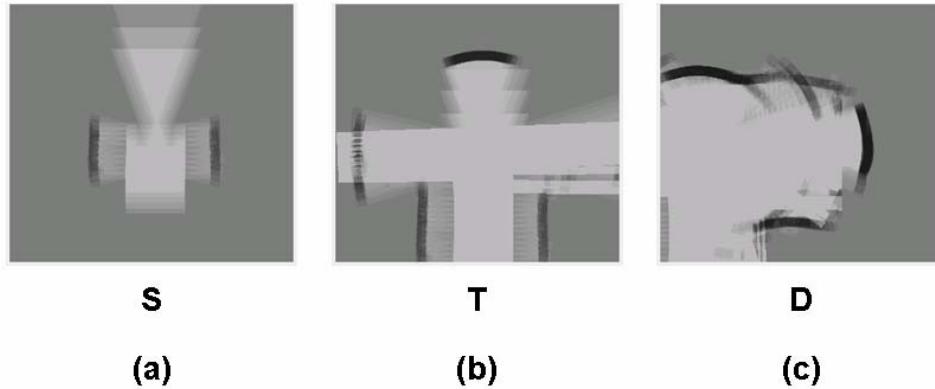


**Figure 3:** Drawing of environment for test described in the Results section. The location of the object of interest for this test, a book, is denoted by the 'X'. The robot positions at the start and end of the test are indicated by the gray triangles (the robot is facing to the top of the figure at the start of the test and to the right of the figure at the end of the test).

At the beginning of the test, the robot was placed in the bottom of the 'T' shown in Figure 3 facing towards the top of the figure. After the agent was initialized, the agent used its follow middle operator in the navigate abstract operator to move toward the top of Figure 3. An 'S' intersection was detected (Figure 4a) and the robot continued toward the top of the figure using the

navigate operator until a ‘T’ intersection was detected (Figure 4b). The intersection was recorded and the agent considered plans of turning left, turning right, and turning around. The turn right plan was selected and the perform-plan operator was used to move the robot through the intersection. At this point, the robot was facing to the right of Figure 3 and began to move to the right using the agent’s navigate operator. After

moving to the right, the ‘D’ intersection was detected (Figure 4c) and recorded and a plan was generated to turn around. While the robot was beginning to perform this plan, the book was identified and recorded, accomplishing the agent’s mission. Figure 5 shows the working memory elements associated with the three intersections and book that were detected during the test.



**Figure 4:** The occupancy grids used to detect the ‘S’ (part (a)), ‘T’ (part (b)), and ‘D’ (part (c)) intersections.

```
(M1 ^intersections M2 ^objects M3) //Identifier M1 has information about intersections & objects
(M2 ^|ID| N16 ^|ID| N9 ^|ID| N2 ^|currentIDnumber| 3) //M2 has information about 3 intersections

(N16 ^type |D| ^|IDnumber| 3 ^|firstDetection| 66 //Intersection N16 is a ‘D’ intersection. It was the third
 ^|firstLocation| L76 ^|lastDetection| 72 // intersection detected (from t=66 to t=72).
 ^|lastLocation| N26)
(L76 ^theta -7.57762 ^x 2.50753 ^y 3.02809) //L76 and N26 contain the robot’s x, y, & theta values at the
(N26 ^theta -9.37133 ^x 2.77526 ^y 2.98862) // first & last locations, respectively, at which the ‘D’ was detected.

(N9 ^type |T| ^|IDnumber| 2 ^|firstDetection| 24 //Intersection N9 is a ‘T’ intersection. It was the second
 ^|firstLocation| L32 ^|lastDetection| 29 // intersection detected (from t=24 to t=29).
 ^|lastLocation| N15)
(L32 ^theta 92.1524 ^x 1.48985 ^y 2.79797) //L32 and N15 contain the robot’s x, y, & theta values at the
(N15 ^theta 91.7937 ^x 1.48143 ^y 3.0074) // first & last locations, respectively, at which the ‘T’ was detected.

(N2 ^type |S| ^|IDnumber| 1 ^|firstDetection| 10 //Intersection N2 is a ‘S’ intersection. It was the first
 ^|firstLocation| L16 ^|lastDetection| 16 // intersection detected (from t=10 to t=16).
 ^|lastLocation| N8)
(L16 ^theta 90. ^x 1.50222 ^y 2.01278) //L16 and N8 contain the robot’s x, y, & theta values at the
(N8 ^theta 90.7175 ^x 1.50401 ^y 2.37497) // first & last locations, respectively, at which the ‘S’ was detected.

(M3 ^|currentObjectIDnumber| 1 ^|objectID| N20) //M3 contains information about 1 object, N20.
(N20 ^|IDnumber| 1 ^|ID| book // N20 is a book. It was the first object detected (from t=69 to t=72).
 ^|firstDetection| 69 ^|firstLocation| L81
 ^|lastDetection| 72 ^|lastLocation| N27)
(L81 ^theta -8.2951 ^x 2.66875 ^y 3.00555) //L81 and N27 contain the robot’s x, y, & theta values at the
(N27 ^theta -9.37133 ^x 2.77526 ^y 2.98862) // first & last locations, respectively, where the book was detected.
```

**Figure 5:** Working memory elements corresponding to intersections and objects detected during test. The text on the left half of the figure is the part of the Soar agent’s working memory associated with information about recorded intersections and objects. The text on the right half of the figure (to the right of the ‘//’ characters) describes the working memory elements located on the same line.

## 5. Summary

A Soar agent has been developed that can search inside a building, find intersections, record them for creation of a topological map, and look for objects of interest using SIFT. Results from a test in a simple environment in which the agent searched for a single object of interest were described. The work described in this paper is related to several research projects mentioned at the beginning of this paper. As in the research described by Kennedy et al. (2007), our work has integrated a mapping capability in a system with a cognitive architecture. However, while the previous research generated a 2-D grid-based cognitive map outside of ACT-R that the ACT-R model could use for spatial reasoning, our work generated topological information within Soar. The use of a local occupancy grid as a source of information for detecting intersections has previously been described by Beeson, Modayil, and Kuipers (2010). Beeson, Modayil, and Kuipers used intersections detected while following a previously prescribed path for exploring an environment to generate the global topological structure of the environment to create an accurate global metric map. While this is a very useful application, our work is interested in using knowledge about each type of detected intersection to make decisions about where to navigate in an environment.

Kelley et al. (2009) have also used SS-RICS to identify common intersection types and make navigation decisions based on knowledge about paths available in each type of intersections. There are several differences between the work described in this paper and their work. While we have currently only implemented a random search strategy, Kelley et al. used a maze solving algorithm to guide their navigation decisions. We have incorporated a probabilistic framework to help with reliable abstraction of symbolic intersection information (Beeson et al., 2010). We have also integrated SIFT into our robot system to recognize objects of interest. Additionally, Kelley et al. discuss how they experienced difficulty in keeping the robot in good positions for autonomous intersection detection and with selecting thresholds to be used in rules that made decisions about the presence of intersections. Our Soar agent uses the operators described in Section 3.2.1 to maximize the likelihood the robot will be in a good position to detect intersections. The use of Hough transforms to estimate the hall width has allowed the fuzzy logic system used to detect intersections to adapt to different width hallways.

There are several improvements that could be made to the system and the results described in this paper. Future tests will be conducted in more challenging environments. Knowledge allowing the Soar agent to

generate a topological map using the recorded intersections will also be added. This topological map can then be used to efficiently return the robot to its location at the beginning of the test. A more detailed Soar agent could be developed that could search more intelligently or attempt to solve the loop closing problem topologically (Beeson et al., 2010). The agent could also take advantage of newer Soar features (J.E. Laird, 2008), such as episodic memory that may be useful to use instead of some of the operators that are used to record information such as robot locations in working memory. Also, additional sensors and sensor processing (e.g., stereo vision for better quality occupancy grids and/or intersection detection, sound localization, the ability to recognize places using SIFT) could be added to the CRS.

## 6. References

- Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*(4), 1036-1060.
- Balaguer, B., Balakirsky, S., Carpin, S., & Visser, A. (2009). Evaluating maps produced by urban search and rescue robots: lessons learned from RoboCup. *Autonomous Robots*, *27*(4), 449-464.
- Beeson, P., Modayil, J., & Kuipers, B. (2010). Factoring the Mapping Problem: Mobile Robot Map-building in the Hybrid Spatial Semantic Hierarchy. *The International Journal of Robotics Research*, *29*(4), 428-459.
- Benjamin, P., Lyons, D., & Lonsdale. (2004). Designing a Robot Cognitive Architecture with Concurrency and Active Perception. In *Proceedings of the AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics*. AAAI.
- Chown, E., Kaplan, S., & Kortenkamp, D. (1995). Prototypes, Location and Associative Networks (PLAN): Towards a unified theory of cognitive mapping. *Cognitive Science*, *19*, 1-51.
- Duda, R. O., & Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Comm. ACM*, *15*(1), 11-15.
- Hanford, S. D., Janrathitikarn, O., & Long, L. N. (2009). Control of Mobile Robots Using the Soar Cognitive Architecture. *Journal of Aerospace Computing, Information, and Communication*, *6*(2).
- Hanford, S. D., & Long, L. N. (2010). Integration of Maps into the Cognitive Robotic System. Presented at the AIAA InfoTech@Aerospace Conference, Washington, DC: AIAA, AIAA

- Paper No. 2010-3350.
- Jones, R., Laird, J., Nielsen, R., Coulter, K., Kenny, R., & Koss, F. (1999). Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine*, 27-41.
- Kelley, T. D., Avery, E., Long, L. N., & Dimperio, E. (2009). A Hybrid Symbolic and Sub-Symbolic Intelligent System for Mobile Robots. In *AIAA InfoTech@Aerospace Conference*. Wahington, DC: AIAA, AIAA Paper No. 2009-1976.
- Kennedy, W., Bugajska, M., Marge, M., Adams, W., Fransen, B., Perzanowski, D., Schultz, A., et al. (2007). Spatial Representation and Reasoning for Human-Robot Collaboration. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence* (pp. 1554-1559). AAAI.
- Kuipers, B., Tecuci, D. G., & Stankiewicz, B. J. (2003). The Skeleton In The Cognitive Map: A Computational and Empirical Exploration. *Environment and Behavior*, 35, 81-106.
- Laird, J. E. (2009). Towards Cognitive Robotics. Presented at the SPIE Defense and Sensing Conferences, Orlando, FL.
- Laird, J. (2008). Extending the Soar Cognitive Architecture. In *Proceedings of the First Artificial General Intelligence Conference*.
- Laird, J., Hucka, M., Yager, E., & Tuck, C. (1991). Robo-Soar: An integration of external interaction, planning, and learning, using Soar. *IEEE Robotics and Autonomous Systems*, 8(1-2), 113-129.
- Laird, J., Newell, A., & Rosenbloom, P. (1987). Soar: An Architecture for General Intelligence. *Artificial Intelligence*, 33(3), 1-64.
- Laird, J., & Rosenbloom, P. (1990). Integrating execution, planning, and learning in Soar for external environments. In *Proceedings of the National Conference of Artificial Intelligence*.
- Long, L. N., Hanford, S. D., & Janrathitkarn, O. (2010). Cognitive Robotics using Vision and Mapping Systems with Soar. Invited Paper, Presented at the SPIE Defense & Security Conference, Orlando, FL.
- Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91-110.
- Mozos, O. M., & Burgard, W. (2006). Supervised Learning of Topological Maps using Semantic Information Extracted from Range Data. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.
- Tomatis, N. (2008). Hybrid, Metric-Topological Representation for Localization and Mapping. In M. E. Jefferies & W. K. Yeap (Eds.), *Robot*

- and Cognitive Approaches to Spatial Mapping* (pp. 43-63). Berlin/Heidelberg: Springer.
- Trafton, J. G., Schultz, A. C., Cassimatis, N. L., Hiatt, L. M., Perzanowski, D., Brock, D. P., Bugajska, M., et al. (2006). Communicating and Collaborating with Robotic Agents. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (pp. 252-278). Cambridge: Cambridge University Press.

## Author Biographies

**SCOTT D. HANFORD** is a graduate student in the Department of Aerospace Engineering at the Pennsylvania State University.

**LYLE N. LONG** is a Distinguished Professor of Aerospace Engineering, Bioengineering, and Mathematics, the Director of the Computational Science Graduate Minor Program, and a Member of the Graduate Program in Neuroscience at the Pennsylvania State University.