Place Recognition and Topological Map Learning in a Virtual Cognitive Robot

Paul R. Smart[1] **and Katia Sycara**[2]
[1]Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ UK
ps02v@ecs.soton.ac.uk
[2]Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
katia@cs.cmu.edu

Abstract—*An ACT-R cognitive model is used to control the spatial behavior of a virtual robot that is embedded in a three-dimensional virtual environment, implemented using the Unity game engine. The environment features a simple maze that the robot is required to navigate. Communication between ACT-R and Unity is established using a network-based inter-operability framework. The ability of the robot to learn about the spatial structure of its environment and navigate to designated goal locations serves as a test of the ability of the framework to support the integrative use of cognitive architectures and virtual environments in a range of research and development contexts.*

Keywords: virtual robotics; virtual environment; cognitive architecture; spatial cognition; spatial memory

1. Introduction

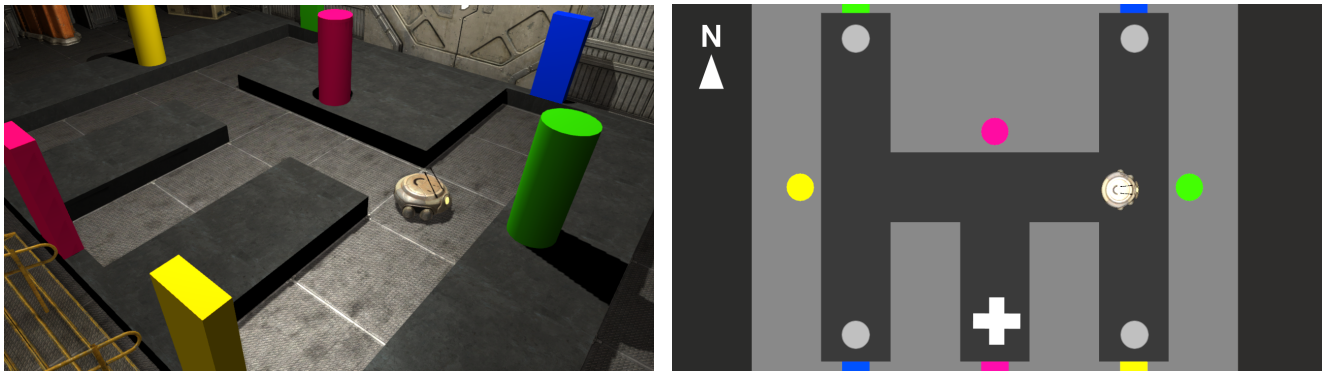
In the effort to develop computational models of cognitive behavior, it often helps to draw on the resources of a reusable framework that incorporates some of the representational structures and computational mechanisms that are assumed to be invariant across multiple cognitive tasks. Cognitive architectures are examples of such frameworks [1]. They can be used to develop models that test ideas relating to the cognitive mechanisms associated with aspects of human performance. In addition, they are sometimes used to implement agents that are capable of performing cognitive tasks or exhibiting signs of behavioral intelligence [2].

Work using cognitive architectures has typically involved the use of computational models that are highly limited with respect to the kinds of agent-world interaction they support. The perceptual inputs to cognitive models are typically very simple, as are the motor outputs. In addition, it is sometimes difficult to precisely simulate the effects that motor outputs have on subsequent sensory input, thereby limiting the extent to which cognitive models can productively incorporate motor actions into ongoing problem-solving processes (see [3]).

One way to address these issues involves the use of virtual environments, such as those encountered in contemporary video games. These can be used to create dynamic and perceptually-rich environments that serve as virtual surrogates of the real world. The cognitive models that are implemented using cognitive architectures can then be used to control the behavior of one or more virtual agents that inhabit these environments. By supporting the exchange of rich bodies of information between the virtual environment and the cognitive model, and by linking the cognitive model to the perceptuo-motor system of a particular virtual agent, it becomes possible to think of cognitive models as being effectively embedded and embodied within a virtual world.

In this paper, we describe a study that combines the use of a cognitive architecture with a virtual environment in order to study the maze learning and place recognition abilities of a virtual cognitive robot. The cognitive architecture used in the study is the ACT-R cognitive architecture [4]. This is one of the most widely used cognitive architectures within the cognitive scientific community. Although the design of ACT-R is inspired by the features of the human cognitive system (e.g., ACT-R consists of a number of modules that are associated with specific cognitive capabilities, such as the memorization and recall of declarative knowledge), it is possible to use ACT-R in the context of research efforts where the aim is not so much the modeling of human cognitive processes as the real-time control of a variety of intelligent systems. This is evidenced by recent work concerning the use of ACT-R in the design of real-world cognitive robots [5]. It is also possible to extend the core functionality of ACT-R in a variety of ways (e.g., via the addition of new modules).

Aside from ACT-R, the focus of the current integration effort is centered on the Unity game engine. This is a game engine, developed by Unity Technologies, that has been used to create a broad range of interactive 2D and 3D virtual environments. Despite its use as a research tool, as well as a platform for game development, the current attempt to integrate ACT-R with Unity is entirely novel: to our knowledge there have been no previous attempts to combine the use of Unity and ACT-R in the context of cognitive agent simulations. The closest approximation to the current integration effort is work by Best and Lebiere [6]. They used ACT-R to control the behavior of humanoid virtual characters in an environment implemented on top of the Unreal Tournament game engine. Our work differs from this previous work in the sense that we are targeting a different



(a) (b)

Fig. 1: Different views of the 'H' Maze environment. The robot is located on the right-hand side of the maze in both images. (a) View from a first-person camera situated external to the maze. (b) View from a top-down tracking camera situated directly above the maze. The tracking camera produces a simplified rendering of the scene in order to support the analysis and visualization of simulation results. The white cross represents the starting location of the robot on all training and testing trials. The grey circles within the maze represent goal locations for the robot during testing trials.

game engine (i.e., Unity) and we do not attempt to control a humanoid virtual character. Instead, we focus on a virtual robotic system that comes equipped with a set of (distinctly non-human) sensor and effector elements. These serve to make the perceptual and behavioral capabilities of the robot unlike those seen in the case of humanoid virtual characters. Another factor that differentiates our work from previous attempts to integrate cognitive architectures with virtual environments concerns our approach to sensor processing. While it is possible to rely on explicit knowledge about the features of virtual world objects (e.g., their position and geometry) as a means of directly calculating important perceptual information (e.g., distance and shape information), the robot in the current study is required to engage in the processing of low-level sensor data as a means of extracting cognitively-useful information from its environment.

As a means of testing the integrity of the ACT-R/Unity integration solution, we rely on the use of a spatial navigation task that requires an ability to (1) recognize spatial locations, (2) learn about the structure of a spatial environment and (3) navigate to specific goal locations. There are a number of factors that motivate the choice of this task in the context of the current work. Firstly, the topic of spatial cognition has been the focus of extensive research efforts in both the robotics and neuroscience communities [7], [8], [9]. This provides a wealth of data and knowledge that can be used to support the development of spatially-relevant cognitive models and associated cognitive processing capabilities. Secondly, spatial navigation is a task that is recognizably cognitive in nature, and it is one that may therefore benefit from the use of a cognitive architecture, such as ACT-R. In addition, the task is of sufficient complexity to require more than just the trivial involvement of the cognitive architecture. In fact, as will be seen below, the task requires the use of multiple existing ACT-R modules, the development of a new custom module, and the exploitation of over 100 production rules. Thirdly, the place recognition component of the task places demands on the perceptual processing capabilities of the robot. This helps to test the mechanisms used for the processing and interpretation of sensor data. Finally, the task requires the continuous real-time exchange of information between ACT-R and Unity in order to ensure that the behavior of the virtual robot is coordinated with respect to its local sensory environment. This serves as a test of the real-time information exchange capabilities of the proposed integration solution.

2. Method

2.1 Environment Design

A simple virtual maze was constructed from a combination of simple geometric shapes, such as blocks and cylinders. The design of the maze is based on that described by Barrera and Weitzenfeld [7] as part of their effort to evaluate bio-inspired spatial cognitive capabilities in a real-world robot. The maze consists of a number of vertically- and horizontally-aligned corridors that are shaped like the letter 'H'. An additional vertically-aligned corridor is used as a common departure point for the robot during training and testing trials (see Figure 1).

A number of brightly colored blocks and cylinders were placed around the walls of the maze to function as visual landmarks. These objects are used by the virtual robot to identify its location within the maze.

2.2 Virtual Robot

The virtual robot used in the current study is based on a pre-existing 3D model available as part of the Robot Lab

project from Unity Technologies. The 3D structure of the virtual robot is defined by a conventional polygonal mesh of the sort typically used in game development. The robot was equipped with three types of sensors in order to support the processing of visual, tactile and directional information. Visual information is processed by the robot's eyes, which are implemented using Unity `Camera` components. For convenience, we refer to these components as 'eye cameras'. The eyes are positioned around the edge of the robot and are oriented at 0°, 90°, 180° and 270° relative to the Y or 'up' axis of the robot in the local coordinate system. This provides the robot with a view of the environment to its front, back, left and right. Given the elevated position of the visual landmarks in the maze (see Figure 1), the eyes were oriented slightly upwards at an angle of 15°. This enabled the robot to see the landmarks, even when it was positioned close to one of the walls of the maze.

In order to keep the visual processing routines as simple as possible, the eye cameras were configured so as to enhance the visibility of the visual landmarks within the scene. In particular, the far clipping plane of each eye camera's view frustum was set to 10 meters. This limited the range of the camera within the scene (although the range was still sufficient to encompass the entire extent of the 'H' Maze, irrespective of the robot's actual position in the maze). The culling mask of each eye camera was also configured so as to limit the rendering of scene objects that were external to the maze environment. Finally, a self-illuminated shader was used for the rendering of visual landmarks by the eye cameras. This shader used the alpha channel of a secondary texture to define the areas of the landmark that should emit light of a particular color. By simply omitting this secondary texture, the visual landmarks had the appearance of objects that emitted light uniformly across their surface. This served to enhance the contrast of the objects (from the robot's perspective) and reduced color variations resulting from different viewing angles. The result of applying these adjustments is shown in the four image insets at the top of Figure 2. These show the view of the maze environment from the perspective of each of the robot's eye cameras.

During the training and testing phases of the experiment (see Section 2.5), the output of each eye camera was periodically rendered to what is known as a `RenderTexture` asset. This is a special type of 2D image asset that captures a view of the virtual environment from the perspective of a particular camera. In essence, each `RenderTexture` asset effectively represents the state of one of the robot's 'retinas' at a particular point in time. The pixel data associated with these images can be processed in order to extract visual features, some of which may indicate the presence of particular objects in the scene. The visual processing routines used in the current study were relatively lightweight and focused on the attempt to detect the brightly colored objects (i.e., the visual landmarks) arrayed around the walls

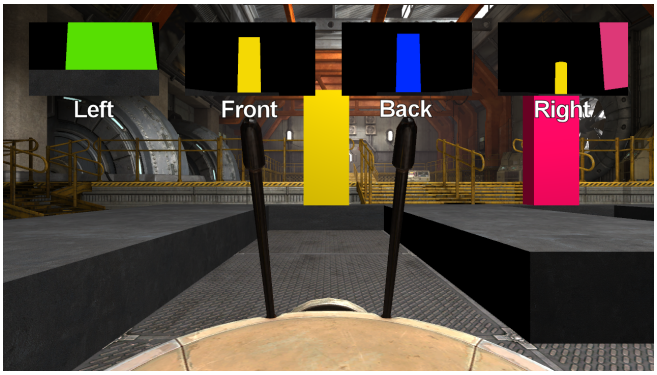


Fig. 2: View of the 'H' Maze from a forward-facing camera situated onboard the robot. The four image insets at the top of the image correspond to the views the robot has of the virtual environment via its eye cameras.

of the maze. These objects were detected by matching the luminance levels of image pixels in the red, green and blue (RGB) color channels to the colors of the objects as they appeared in the robot's eye cameras. A custom `RobotEye` component was developed to support the design-time configuration of the eye cameras with respect to the detection of the visual landmarks. This component supports the specification of target colors that should be detected by each eye camera during the post-rendering analysis of each `RenderTexture` asset. The component also provides access to two properties that control the sensitivity of the robot's eye cameras. These are the 'tolerance' and 'threshold' values. The tolerance value represents the range of luminance levels in each color channel that is recognized as a match to the target luminance level. A value of 0.01, for example, means that deviations of ± 0.01 from a target luminance level (in each color channel) will be recognized as a match to the target color[1]. The threshold value specifies the minimum number of matching pixels that must be present in the image in order for the `RobotEye` component to signal the detection of a particular color. For the purposes of the current study, the tolerance value was set to a value of 0.01 and the threshold value was set to a value of 1500. In addition, each retina was sized to 200×200 pixels to give a total of 40,000 pixels per eye camera on each render cycle.

In addition to the eye cameras, the robot was also equipped with 'whiskers' that functioned as tactile (or proximity) sensors. The aim of these sensors was to detect the presence of maze walls in the forward, left, right and backwards directions. The whiskers extended outwards from the robot's body in the same directions as the eye cameras and were of sufficient length to detect when the robot was adjacent to a maze wall. This enabled the robot to detect

[1]In Unity, the values of RGB channels range from 0 to 1, so a value of 0.01 represents 1% of the total value range.

the presence of particular situations, such as when it was in a corridor (e.g., the left and right whiskers were both in contact with maze walls) or when it had reached the end of one of the maze arms (e.g., the forward, left and right whiskers were all in contact with maze walls). The information provided by the whiskers assists in helping the robot to localize itself within the maze. The whiskers also function to provide affordances for action, helping the robot to decide when it needs to turn and what directions it can move in. From an implementation perspective, the whiskers were implemented using ray casting techniques: each time the robot was required to report sensory information to ACT-R, rays were projected from the robot's body and any collisions of the rays with the walls of the maze were recorded.

The final sensor used by the robot was a directional sensor. This functioned as an onboard compass. The sensor reading was based on the rotation of the robot's transform in the world coordinate system. A rotation of 0° thus corresponded to a heading value of 'NORTH'; a rotation of 90°, in contrast, corresponded to a heading value of 'EAST'.

For the purposes of this work, the directional movement of the robot was restricted to the north, south, east and west directions: these are the only directions that are needed to fully explore the 'H' Maze environment. The robot was also capable of making rotational movements to orient itself in the north, south, east, and west directions. Turning movements were implemented by progressively rotating the robot's transform across multiple update cycles using spherical linear interpolation techniques. Linear movements, in contrast, were implemented by specifying the velocity of the robot's `Rigidbody` component, a component that enabled the robot to participate in the physics calculations made by Unity's physics engine. Both movements occurred in response to the instructions received from an ACT-R model, and in the absence of this input, the robot was behaviorally quiescent.

2.3 Cognitive Modeling

The cognitive modeling effort involved the development of an ACT-R model that could support the initial exploration of the maze and the subsequent navigation to target locations. The requirements of the model were the following:

1) **Motor Control:** The model was required to issue motor instructions to the robot in response to sensory information in order to orient and move the robot within the maze.
2) **Maze Learning:** The model was required to detect novel locations within the maze and memorize the sensory information associated with these locations.
3) **Route Planning:** The model was required to use the memorized locations in order to construct a route to a target location.
4) **Maze Navigation:** The model was required to use route-related information in conjunction with sensory feedback in order to monitor its progress towards a target location.

In addition, in order to analyze the structure of the robot's spatial memories and compare navigational performance under different test conditions, it was important for the model to be able to serialize and deserialize memorized information to a persistent medium.

The ACT-R model developed for the current study consists of 126 production rules in addition to ancillary functions that control the communication with Unity (see Section 2.4). A key goal of the model is to memorize spatial locations that are distinguished with respect to their sensory properties (i.e., unique combinations of visual and tactile information). These locations are referred to as 'place fields' in the context of the model. Each place field is created as a chunk in ACT-R's declarative memory, and retrieval operations against declarative memory are used to recall the information encoded by the place field as the robot moves through the maze. The collection of place fields constitutes the robot's 'cognitive map' of the maze (see [9]). This map is structured as a directed graph in which the place fields act as nodes and the connections between the nodes are established based on the directional information that is recorded by the robot as it explores the maze. Any two place fields that are created in succession will be linked via a connection that records the direction the robot was moving in when the connection was made. For example, if the robot creates a place field (*PF1*) at the start of the simulation and then creates a second place field (*PF2*) while heading north from the start location, a connection will be established between *PF1* and *PF2* that records *PF1* as the source of the connection, *PF2* as the target of the connection and 'NORTH' as the direction of the connection. The cognitive map, as the term is used in the current study, is thus a representational structure that encodes information about the topological relationships between place fields based on the exploration-related movements of the virtual robot.

The productions of the ACT-R model were used to realize the motor control, maze learning and navigation functions mentioned above; the route planning function, however, was implemented using separate Lisp routines. In order to plan a route, the robot first needs to be given a target location. This was specified at the beginning of trials that tested navigational performance (see Section 2.5). The robot then needs to identify its current location within the maze. The robot achieved this by comparing current sensory information with that stored in memory (in the form of place field representations). Finally, the robot needs to compute a sequence of place fields that encode the path from the start location to the target location. This was achieved via the use of a spreading activation solution that operated over all the place fields in the robot's cognitive map (i.e., the contents of the

robot's spatial memory). The spreading activation solution involved the initial activation of the place field corresponding to the robot's start location, and this activation was then propagated to neighboring place fields across successive processing cycles until the place field representing the target location was finally reached. The chain of activated place fields from the start location to the target location specifies the sequence of place fields (identified by combinations of sensory information) that must be detected by the robot as it navigates towards the target. Importantly, the connections between adjacent place fields in the computed route serves to inform the robot about the desired direction of travel as each place field is encountered. For example, if the connection between the first and second place fields in the route has an associated value of 'NORTH' and the robot is currently facing north, then the model can simply instruct the robot to move forward. If the robot is facing south, then the robot needs to implement a 180° turn before moving forward.

In order to avoid situations where the robot failed to detect successive place fields in the planned route (either as a result of delays in sensor feedback or the close proximity of topologically-adjacent place fields), the robot attempted to match received sensor information to *all* route-related place fields every time new sensor information was received. This enabled the robot to continually monitor its progress against the planned route and avoid confusion if some locations in the route were over-looked.

An initial pilot study using an earlier ACT-R model (see [10]) revealed a tendency for errors to sometimes occur in navigation-related decisions. Although this did not affect the ability of the robot in the pilot study to ultimately reach a particular goal destination, it did lead to inefficiencies in navigational behavior. An analysis of the structure of the robot's spatial memory in the context of this earlier study revealed that the problem originated from a failure to adequately discriminate between spatially-distinct locations during maze learning. Given the robot's perceptual capabilities, some of the locations in the maze can appear identical, and this can lead to situations where erroneous linkages are created between non-adjacent place fields. The result is a breakdown in the extent to which the cognitive map provides a faithful representation of the actual topological structure of the environment. In order to address this shortcoming, the current cognitive model attempted to categorize visual inputs based on the number of pixels of a particular color that were contained in the image generated by each eye camera. Pixel counts between 1500 and 6000 (for a particular color) were thus categorized as indicating the presence of 'small' colored objects, and pixel counts above 6000 were categorized as indicating the presence of 'large' colored objects[2]. The addition of this admittedly simple categorization scheme was sufficient to yield adequate discriminative capabilities in the context of the 'H' Maze; it is likely, however, that more refined schemes will be required in the case of more complex spatial environments.

[2]The detection threshold of the eye cameras was equal to 1500, so pixel counts below this value were treated as equal to zero.

2.4 ACT-R/Unity Integration Solution

In order for the ACT-R model to control the movements of the virtual robot in response to sensory information, it is necessary for the ACT-R environment and the Unity game engine to engage in bidirectional modes of communication. This is problematic because Unity is implemented in C++, while ACT-R is implemented in Lisp. In addition, the need to run Unity and ACT-R in parallel can place significant demands on the processing and memory resources of the host machine, and this can undermine the real-time responsiveness of both systems.

As a means of addressing these concerns, we developed a network-based solution to support the integration of ACT-R with the Unity game engine. The solution is based on an existing approach to integrating ACT-R with external environments that goes under the heading of the JSON Network Interface (JNI) [11]. The JNI enables ACT-R to exchange information with a variety of external environments using a combination of a TCP/IP connectivity solution and messages formatted using the JavaScript Object Notation (JSON) data interchange format. In order to make use of this approach in the context of environments built on top of the Unity game engine, we developed a set of components collectively referred to as the ACT-R Unity Interface Framework [10]. These components provide support for the automatic handling of connection requests made by ACT-R models using the JNI. They also enable Unity-based virtual characters to send information to specific ACT-R models and respond to ACT-R commands. The result is a generic solution for enabling ACT-R models to control the behavior of virtual characters in any Unity-based virtual environment (either 2D or 3D). By combining the framework with the JNI, we were able to run ACT-R and Unity on different machines (thus addressing performance issues) and establish bidirectional forms of communication between the two systems using a client-server model (with the ACT-R model acting as the client and Unity acting as the server). Further details of the integration solution can be found in Smart et al. [10].

At runtime, sensor information from the virtual environment was periodically posted to ACT-R as part of a 'sensor processing cycle'. For performance reasons, this was constrained to run at a frequency much lower than that of the game engine's main update loop (a frequency of 2Hz was used in the current study). During each sensor processing cycle, information from all of the robot's sensors was posted to ACT-R using a single JSON-formatted message. The ACT-R model received this information and responded to it by issuing motor commands that were posted back to Unity (again as JSON-formatted messages). These motor commands were themselves generated by a sequence of

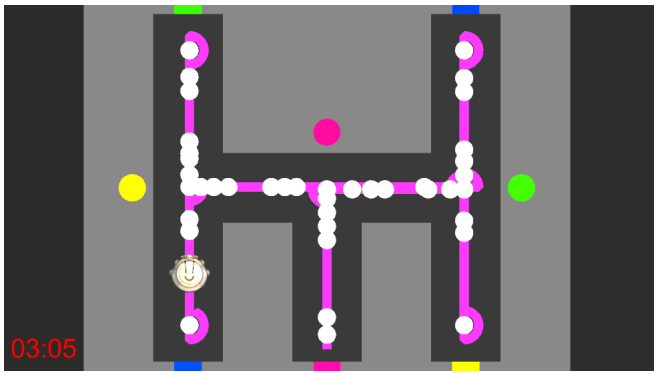


Fig. 3: A cognitive map of the environment formed during one of the training trials of the experiment. Each white circle symbolizes a place field that was created by the robot as it explored the maze. The place fields correspond to nodes in a topological map of the environment.

production firings corresponding to the cognitive processing steps implemented by the ACT-R model. On receipt of the motor commands, the Unity game engine dispatched the commands to the virtual robot, which then assumed responsibility for the actual implementation of motor actions.

2.5 Procedure

In order to test the integrity of the ACT-R/Unity integration solution, as well as the performance of the cognitive model, we performed a simple experiment involving a series of simulations. Each simulation consisted of two phases: a training phase and a testing phase. In the training phase, the robot was allowed to move around the maze and form a cognitive map based on its experiences. Once the robot had explored all of the maze, the training phase was terminated and the robot's cognitive map was saved to disk. In the subsequent testing phase, the cognitive map was loaded into declarative memory and the robot was given a series of target locations to navigate towards. These target locations were situated at the ends of each of the vertical corridors comprising the long arms of the 'H' Maze. The starting location of the robot was the same in all testing and training trials (see Figure 1b).

The simulation was repeated a total of five times in order to test the reliability of the model and the integrity of the ACT-R integration solution. This resulted in a total of five cognitive maps that were acquired on five separate training trials. It also resulted in data from (4 $\times$ 5) 20 testing trials that highlighted the navigational performance of the robot.

3. Results

The structure of one of the cognitive maps formed during one of the training phases of the experiment is shown in Figure 3. The white circles in this figure indicate the position of the place fields that were formed by the robot as it moved around the maze. The magenta trail represents the path of the robot and indicates the extent of the robot's exploratory activity.

Table 1: Table showing mean and standard deviation values for key dependent variables. Data was obtained from 5 simulations using identical conditions and parameters.

Dependent Variable	$\bar{X}$	σ
Training phase duration (seconds)	194.80	11.63
# place fields	41.00	2.45
# place field connections	44.60	3.13
Time to top-left target (seconds)	45.20	1.10
Time to bottom-left target (seconds)	45.40	1.52
Time to top-right target (seconds)	46.40	1.14
Time to bottom-right target (seconds)	46.40	1.95
# ACT-R messages (per minute) across all trials	123.66	1.78
# Unity messages (per minute) across all trials	274.07	4.69

Figure 4 shows the path followed by the robot as it navigated to one of the target locations (situated at the top left of the maze) in one of the testing phases of the experiment (the cognitive map, in this case, is the same as that shown in Figure 3). The robot was able to successfully navigate to each of the target locations in all test-related trials of the experiment. In addition, unlike the results that were obtained in an earlier pilot study (see [10]), the navigational performance of the robot was highly efficient, with no detours being made by the robot *en route* to the target locations. Table 1 summarizes some of the key results of the study. In addition, a video showing the behavior of the robot during the training and testing phases of the experiment is available for viewing from the YouTube website[3].

4. Conclusion

This study has shown how the ACT-R cognitive architecture can be used to control the behavior of a virtual robot that is embedded in a simulated 3D environment. A key aim of the study was to test the integration of ACT-R (which represents one of the most widely used cognitive architectures) with the Unity game engine (which represents one of the most widely used game creation systems). The integration solution builds on an existing approach to integrating ACT-R with external environments [11] in order to support bidirectional modes of information exchange between an ACT-R model and a Unity-based virtual environment. The two systems were hosted on separate machines during the course of the simulations, a strategy that serves to distribute the computational overhead associated with running both systems at the same time.

The task chosen to test the integration solution was a spatial navigation task that required an ability to learn about the spatial structure of a virtual 3D environment, recognize specific locations within the environment based

[3]See `http://youtu.be/IpoReu_PV3M`

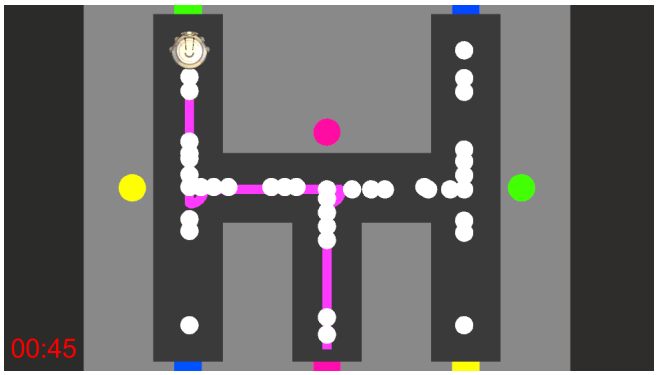


Fig. 4: The path taken by the robot to reach one of four target locations during one of the testing phases of the experiment.

on local perceptual information, and countenance behavioral responses based on a combination of local sensory cues, spatial knowledge and navigation-related goals. The ACT-R model developed to support these capabilities relied on a combination of visual, tactile and kinesthetic information in order to create memorial representations encoding the topological structure of the spatial environment. This approach resembles that seen in the case of real-world robotics research (e.g., [12]), and it is also consistent with the idea of visual and kinesthetic information being used to construct cognitive maps that subsequently guide the navigational behavior of a variety of animal species [9].

One extension of the current work could aim to improve our understanding of the cognitive mechanisms that are sufficient to yield adaptive navigational responses in other kinds of spatial environment. An important focus of attention, here, concerns the ability of virtual robots to exhibit navigational competence in the kinds of mazes that are typically encountered in bio-behavioral research (e.g., the radial-arm maze [13] and the Morris water maze [14]). This could establish the basis for cognitive models that attempt to emulate the spatial behavior of human and non-human subjects under specific test conditions.

Another potential target for future work concerns the enrichment of the cognitive representations used by the ACT-R model to support more sophisticated forms of spatial reasoning and behavioral control. One example here concerns the integration of metric information (e.g., information about angles and distances) into the topological map representation. Such information is deemed to be an important element of the spatial behavior of animals, and it is typically the focus of perceptual processing in the case of biologically-inspired robotic models of spatial navigation ability [8].

Future work could also aim to address some of the sensory and motor limitations of the robot used in the current study (recall the steps taken to simplify the visual processing of `RenderTexture` assets in Section 2.2). This includes work to improve the sophistication of visual processing capabilities, perhaps using techniques derived from computer vision research.

Finally, the availability of the current ACT-R/Unity integration solution opens up a range of relatively new research opportunities. One of these concerns the use of the integration solution to perform computational simulation studies that are relevant to current theoretical and empirical work in embodied, situated and extended cognitive science. Crucially, these simulations could serve as an important adjunct to studies that attempt to evaluate the role that environmentally-extended processing loops (and issues of material embodiment) play in the realization of human-level cognitive capabilities (see [3]).

References

[1] P. Thagard, "Cognitive architectures," in *The Cambridge Handbook of Cognitive Science*, K. Frankish and W. M. Ramsey, Eds. Cambridge, UK: Cambridge University Press, 2012, pp. 50–70.

[2] J. Rickel and W. Lewis Johnson, "Task-oriented collaboration with embodied agents in virtual worlds," in *Embodied Conversational Agents*, J. Cassell, J. Sullivan, and S. Prevost, Eds. Cambridge, Massachusetts, USA: MIT Press, 2000.

[3] A. Clark, *Supersizing the Mind: Embodiment, Action, and Cognitive Extension*. New York, New York, USA: Oxford University Press, 2008.

[4] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," *Psychological Review*, vol. 111, no. 4, pp. 1036–1060, 2004.

[5] U. Kurup and C. Lebiere, "What can cognitive architectures do for robotics?" *Biologically Inspired Cognitive Architectures*, vol. 2, pp. 88–99, 2012.

[6] B. J. Best and C. Lebiere, "Cognitive agents interacting in real and virtual worlds," in *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Interaction*, R. Sun, Ed. New York, New York, USA: Cambridge University Press, 2006.

[7] A. Barrera and A. Weitzenfeld, "Bio-inspired model of robot spatial cognition: Topological place recognition and target learning," in *International Symposium on Computational Intelligence in Robotics and Automation*, Jacksonville, Florida, USA, 2007.

[8] N. Burgess, J. G. Donnett, K. J. Jeffery, and J. O'Keefe, "Robotic and neuronal simulation of the hippocampus and rat navigation," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 352, no. 1360, pp. 1535–1543, 1997.

[9] B. Poucet, "Spatial cognitive maps in animals: New hypotheses on their structure and neural mechanisms," *Psychological Review*, vol. 100, no. 2, pp. 163–192, 1993.

[10] P. R. Smart, T. Scutt, K. Sycara, and N. R. Shadbolt, "Integrating ACT-R cognitive models with the Unity game engine," in *Integrating Cognitive Architectures into Virtual Character Design*, J. O. Turner, M. Nixon, U. Bernardet, and S. DiPaola, Eds. Hershey, Pennsylvania, USA: IGI Global, in press.

[11] R. M. Hope, M. J. Schoelles, and W. D. Gray, "Simplifying the interaction between cognitive models and task environments with the JSON network interface," *Behavior Research Methods*, vol. 46, no. 4, pp. 1007–1012, 2014.

[12] M. J. Mataric, "Navigating with a rat brain: A neurobioiogicaliy-inspired model," in *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behaviour*, J.-A. Meyer and S. W. Wilson, Eds. Boston, Massachusetts, USA: MIT Press, 1991.

[13] D. S. Olton and R. J. Samuelson, "Remembrance of places passed: Spatial memory in rats," *Journal of Experimental Psychology: Animal Behavior Processes*, vol. 2, no. 2, pp. 97–116, 1976.

[14] R. G. M. Morris, "Spatial localization does not require the presence of local cues," *Learning and Motivation*, vol. 12, no. 2, pp. 239–260, 1981.