REACT-R and Unity Integration "Regular Research Paper"

Llewyn Salt[1,3], **Julian Wise**[2,3], **Charlotte Sennersten**[3], **and Craig A. Lindley**[3]
[1]ITEE, University of Queensland, Brisbane, Queensland, Australia
[2]CSIT, RMIT, Melbourne, Victoria, Australia
[3]Data61, CSIRO, Hobart, Tasmania, Australia

Abstract— *This paper presents REACT-R which is a modification of an existing cognitive architecture, ACT-R, to incorporate robot embodiment. Robot embodiment is facilitated by situating the cognitive architecture within the robot and allowing it to interact with the environment through its actuators and sensors. The REACT-R module offers flexibility by using a UDP connection to integrate with the Robot Operating System (ROS) allowing REACT-R to be used in simulations or on physical models. We have successfully integrated REACT-R with an interactive 3D Game Engine to autonomously control a simulated quadrotor flying vehicle. The cognitive model also contains varying levels of autonomy, providing a human pilot with the option of controlling different aspects or levels of robot operation.*

Keywords: UAV, Robotics, ACT-R, Cognitive Architectures, Artificial Intelligence

1. Introduction

Following great advances in robotics technology, there has been increased interest in developing cognitive robotics. The availability of systems like the Robot Operating System (ROS) [1] and perception and action systems like Tekkotsu [2] have made it possible for researchers to develop scalable levels of robotic control, from highly manual to fully automatic. Details of low level tasks can be abstracted away, such as moving each of a set of joints individually or rotating a motor to "pick up a chess piece" [3]. This allows AI researchers to concentrate upon higher level decisions rather than being bogged down in lower level path optimisation or fine motor control.

Robotics research has typically focused on optimising processes to perform specific low level tasks and integrating these lower level actions to create robots that can operate well within small problem spaces. Cognitive science and AI deal with more generalised tasks that are open-ended, knowledge intensive, and span longer time intervals [3].

Robotic optimisation processes rely on computations rather than memory for processing power and were shaped this way by the fast CPUs but slow BUS speeds available to access memory in standard von Neumann computer architectures [3]. Conversely, biological brains are capable of performing robust computations from large amounts of memory and computing elements that are slow, inhomogeneous and faulty, taking advantage from having memory and computation happening in the same location to mitigate the bottleneck found in von Neumann machines [4].

Cognitive architectures attempt to facilitate the creation and understanding of agents that have the same capabilities as humans, which is a central goal of both AI from a cognitive perspective and cognitive science [5]. Cognitive architectures like ACT-R or Soar were initially created to model human cognition but have been expanded to work in AI for simulated environments or on robots [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16].

Cognitive embodiment as a concept focuses on the limitations, affordances and features of a physical embodiment of a cognitive processor[17]. Embodied cognition is a theory that emphasises that conceptual representations that constitute our knowledge are dependent on our sensory and motor experiences.

Within the context of this paper, ACT-R is provided with a physical embodiment in the form of a UAV both within a simulated environment in a commercial 3D visualisation and physics engine, and communicating via software operating as a ROS node on a physical quadrotor. In this case, the simulation mirrors the physical UAV.

1.1 Motivations

There are many challenges faced with directly applying a cognitive model to robot embodiment, especially associated with the relationship between high level symbolic models of cognition and the meaning of computations over these models for a physical robot situated in the physical world. In particular, high level symbolic models of a verbal/textual kind have arbitrary or conventional associations between the tokens in which they are expressed and the references and meanings of those expressions that originate in the human use of tokens as signs. This means that the expressed rules do not have any intrinsic meaning for an artificial cognitive agent There are many challenges faced with directly applying a cognitive model to robot embodiment, especially associated with the relationship between high level symbolic models of cognition and the meaning of computations over these models for a physical robot situated in the physical

world. In particular, high level symbolic models of a verbal/textual kind have arbitrary or conventional associations between the tokens in which they are expressed and the references and meanings of those expressions that originate in the human use of tokens as signs. This means that the expressed rules do not have any intrinsic meaning for an artificial cognitive agent /citeLindley2013; the meaning must be engineered into the system by establishing functional relationships between symbolic structures representing rules and what those structures mean in the embodied context. That is, the relationships between symbolic expressions and sensor inputs and actuator outputs must be explicitly engineered. This results in a three-strand development process: i) crafting rules that make sense in terms of the deductive and procedural functions that they are intended to have for a human author, ii) creating the lower level sensor, behaviour and control functions of the physical platform, and iii) creating the links from high level symbolic expressions through the engineered system of embodiment that can assure correct denotational and functional meanings in the situated and embodied operation of the robot in the physical world. Hence, rule sets need to be planned, programmed, and tested over multiple iterations to ensure successful operation and management of sensory information being input into the cognitive system. Inputting sensory information requires fitting the electronics to the embodied cognitive model and fine-tuning the electronic components, both of which add layers of complexity to the cognitive modelling process in the management of electrical signals, adding more areas of potential failure to the testing and debugging process.

In the context of the project desribed in this paper, direct development of autonomous rule sets within the physical body confines the potential expression of productions to the behavioural space of the physical body, and the levels of abstraction provided for interfacing with the physical system. For example, the propellers are required to be perfectly tuned with all the sensors working correctly before the cognitive model could perform any missions on the drone. This means that development has an inherent bottleneck for the cognitive architecture where the body needs to operate perfectly up to the level of its control interface before cognitive productions can be tested in the embedded agent.

Using a simulation environment can aid the development process by allowing development and testing of the cognitive level in parallel with developing the platform for its physical embodiment.

Having an emulated virtual embodiment operating within a simulation engine means that complex rule-sets can be programmed and tested without the risk of bugs within the hardware development slowing down the development process. Additionally, more elaborate and 'riskier' rule sets can be developed, performed and fine-tuned without risking damage to the cognitive architecture's physical embodiment. For example, whilst establishing a rule set for UAV positional control, the height control may malfunction during development, causing the physical drone, which is inherently fragile, to fall multiple times in the development process. Once the rule sets have been successfully developed and tested within the simulation, the same rule set is applied to the physical embodiment with minimal modifications required and with a high degree of confidence that the cognitive architecture's logic is sound.

The ability to test the cognitive architecture operating on a simulated model of physical embodiment prior to deployment into the physical embodiment minimizes bottlenecks between development of the cognitive architecture and its physical platform. The potential for developers to be productive in programming more elaborate rule sets for the cognitive architecture, without the risk of bugs within the electronic circuitry or damaging the physical embodiment with untested rule sets. Such a development process shifts the emphasis to more complex cognitive products and allows for more elaborate rule sets to be tested in a virtual environment before being deployed onto the physical model.

2. Overview

Fig. 1 shows the REACT-R module integrating with ROS which can then connect to the 3D/physics simulation or to a UAV or other robot. The decision to use ROS means that the REACT-R module has high portability and can be used with existing robotics platforms. It also allows for users to have access to the many of open-source ROS modules available.

This paper reports the integration of the REACT-R module with the 3D platform simulation as a proof of concept. Future work will involve porting this model to the UAV for operating in an underground mine environment (see [**?**]).

2.1 Cognitive Architecture

The rise of cognitive architectures emerged from Allen Newell's attempts to develop a Unified Theory of Cognition (UTC)[18], [19], defining the hallmark characteristics which underlie the representation of cognition. The UTC aims to define what a cognitive architecture includes and how it relates to human cognition. Essentially it abstracts human cognition into subsections that allows models to be created and evaluated. Furthermore it seeks to explore what it means to be cognitive and the follow on effects it has on problem solving and decision making [3].

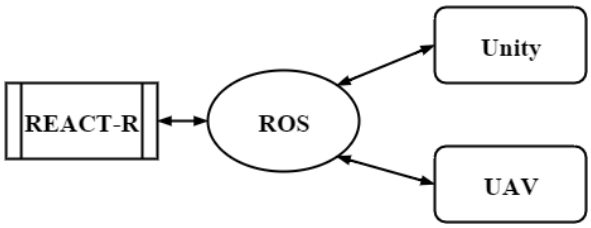

Fig. 1: System Overview of REACT-R with Simulation

The cognitive architecture provides a top-down structure of cognition, where manual robotic refinements can be abstracted into decisions, which the cognitive architecture can manually select.

3. Cognitive Architecture

3.1 ACT-R

ACT-R was developed by John Anderson at Carnegie Mellon University to model human cognition and the mechanism by which humans recall information and solve problems given the current state of the buffers and its procedural and declarative memory [20]. Declarative memory includes facts like *'Sydney is in Australia'*. Procedural knowledge is the how of things, like how you would grasp a cup or how you brush your teeth. The buffers are essentially how ACT-R interacts with the world. ACT-R has visual, aural, goal, motor, speech, imaginal and retrieval buffers[21]. As with any agent ACT-R requires a goal it wishes to fulfill. The goal buffer contains the current goal. The data type in ACT-R is chunks, which act like structures containing information required by the buffers. ACT-R takes an appropriate action based on its goal and buffer values.

ACT-R learns on a structural and statistical level. That is, the activation of declarative chunks will increase based on how often they are used by productions. The cost and success of productions are updated based on their observed behavior. Productions can also merge through repeated sequences of productions to create new ones [5].

ACT-R was designed to emulate human users on a computer to emulate various cognitive tests. This means that the buffers were designed for a 2D screen and a keyboard and mouse. This makes it difficult to utilise ACT-R in its default form for robot AI. Fortunately, ACT-R was designed to be modular and easily modifiable, making it relatively easy to customise ACT-R.

To this end ACT-R/E was created to place the cognitive architecture into a physical embodiment with the goal of modeling human sensory phenomena. The cognitive architecture can accordingly have sensory phenomena as inputs into the cognitive framework from an external environment which would be taken into account during the decision making process [8].

ACT-R/E is not open source and it still incorporates many buffers that could potentially make customisation difficult.

3.2 REACT-R

The novel REACT-R variant reported in tis paper adds a module to the ACT-R 7.0 core software package to allow for a robot embodiment of ACT-R, Fig. 2 shows the REACT-R software architecture. ACT-R has been expanded upon to extend it from an abstraction of a human mind interacting with a screen using a keyboard and mouse [22] to a model that can function as a robot embodiment of cognition with any number of sensors. In this instance the embodiment was a virtual UAV sitting in a rendered model of a real mine. The mine map was formatted as a .dxf file and then voxelised, i.e. turned into multiple atomic cubes.

The UAV sensors consisted of:

- Four Sonar Sensors
- An Optical Flow Sensor
- An 6 DOF Inertial Motion Sensor

The REACT-R module takes in the sonar and optical flow sensory information and leaves the lower level control of attitude and position to the UAV control system.

REACT-R was designed to be intuitive and modular. It is easy to integrate with robotics projects that are currently using ROS. Fig. 1 shows how REACT-R can be interchangeable between a simulation and a robot.

REACT-R uses only one additional buffer that communicates using a UDP connection via ROS nodes. The buffer can be easily modified to incorporate any sensor configuration and send back any commands. ACT-R/E has visual, aural, vocal, motor, configural, and manipulative modules to emulate human cognition, as the quadrotor model requires sensory inputs which are not human-centric (sonar, lidar and optical flow). Communication through the REACT-R module was seen as preferable providing a centralised and modifiable implementation. This module consists of two buffers: Sensors and Actuators. The inputs can be any number of sensors and the outputs can be whatever is required: In our case it outputs the direction the vehicle should travel in.

The REACT-R module is designed to work as a co-pilot with a human and augment the user experience. To this end we have created the REACT-R module with varying levels of autonomy:

- None - REACT-R does not interfere at all with the pilot's flight.
- Semi - REACT-R provides hover control for the UAV and will override user control if sonar sensors reading less than a nominated distance from a surface.
- Functional - REACT-R will move autonomously to a goal location avoiding obstacles as it goes. The operator is still able to take control if they believe REACT-R is not operating as desired.

REACT-R operates as a high level decision maker, being capable of perceiving phenomena and performing a decision making process. Given the architectural topology of REACT-R within its environmental context, commands can be output through the module to a lower level language operating on the robot embodiment platform.

The newly created REACT-R module does not discriminate between whether the Cognitive Architecture inhabits a simulated environment, such as our 3D simulator, or whether the cognitive processor exists in the physical world relating to its surrounding environment over the module connection to ROS.

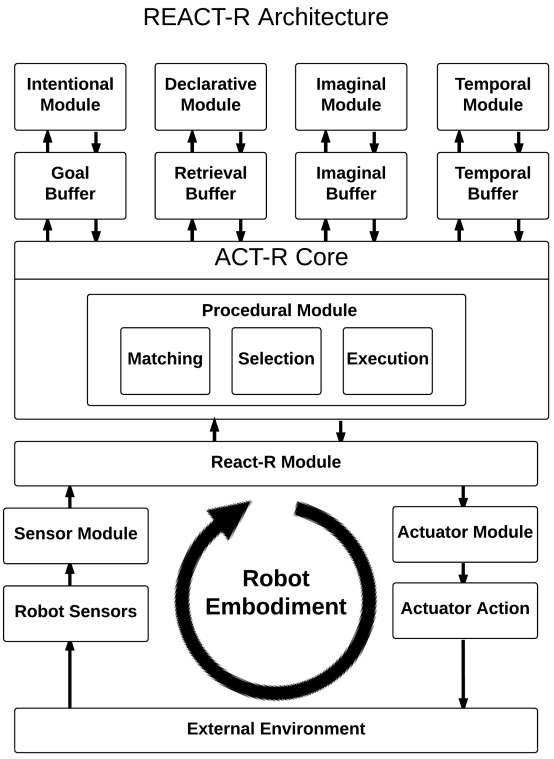

Fig. 2: Software Architecture Diagram for REACT-R Integration

4. Simulation

We created a simulation of a UAV to test the REACT-R module. This was done using Unity, a 3D game engine that uses NVidia's PhysX physics calculations.

The UAV was modelled in BlendR and then put in the Unity simulation. Forces are added depending on inputs to influence UAV motion. The sensors are also simulated by taking measurements that would be provided by these sensors using either ray casting or rigid body mechanics.

A Computer-aided design (CAD) mesh rendering of an underground mine was imported into Unity as a simulated environment. A UAV was modelled with kinematics as an object within the simulated environment for ACT-R to embody.

The simulated embodiment of ACT-R is placed within a surrounding environment with virtual sensors programmed on the UAV within Unity to provide an input to ACT-R of spatial sensory recognition, passed through the REACT-R module.

For example, if the operator is operating in semi or functional autonomy, sensors on the UAV will detect the distance that it is from other objects and this data will be made available to the REACT-R module and subsequently the ACT-R productions.

Once the variable and metric value has been passed through UDP to the ACT-R cognition, it provides a basis for sub-decision making in the process of achieving goals assigned by a human. For example, within an environmental survey mission, a rule set may be established within ACT-R to avoid collisions and to move around physical bodies.

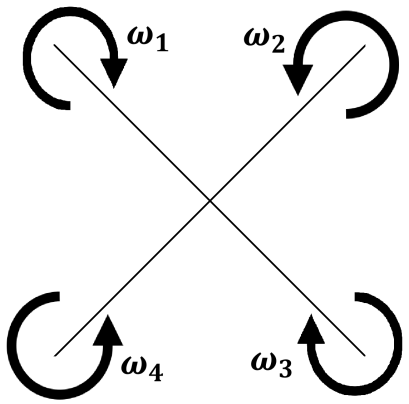

Fig. 3: X Configured Quadrotor with Propeller Labeling

Within the context of a Unity environment, the kinematics and sensory inputs of the UAV were programmed C#, using a UDP connection to REACT-R for the top level decision making process. REACT-R receives relational positional information and tells a positional control system how to move in the virtual space.

4.1 Kinematics

It was desired that the UAV be as realistic as possible, therefore a kinematics model was created. This was done by relating the roll, ϕ, pitch, θ, yaw, ψ, and desire thrust to motor speeds. To be at equilibrium, stable hovering, the kinematics must satisfy these equations:

$$\sum_{i=0}^{4} T_i = -mg \tag{1}$$

$$T_{1-4} || g \tag{2}$$

$$\sum_{i=0}^{4} M_i = 0 \tag{3}$$

$$(\omega_1 + \omega_3) - (\omega_2 + \omega_4) = 0 \tag{4}$$

Where m is the mass of the UAV, g is gravity, and T_i represents the thrust generated, M_i is the moment generated, and ω is the rotational velocity of each propeller, Fig. 3 shows which the X configuration and the labeling convention of each propeller.

To move the UAV these equilibrium values must be disturbed. That means that the sum of the thrust is greater than $-mg$ to go up and less than $-mg$ to go down.

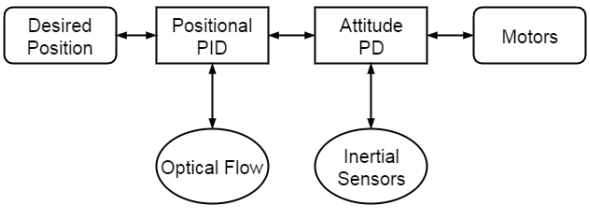

Fig. 4: Control Flow

To obtain the rotational rates we calculate:

$$\dot{\psi} = k_Y((\omega_1 + \omega_3) - (\omega_2 + \omega_4)) \tag{5}$$

$$\dot{\phi} = k_R((\omega_1 + \omega_4) - (\omega_2 + \omega_3)) \tag{6}$$

$$\dot{\theta} = k_P((\omega_1 + \omega_2) - (\omega_3 + \omega_4)) \tag{7}$$

$$F = k_F(\omega_1 + \omega_2 + \omega_3 + \omega_4) \tag{8}$$

Where k_Y, k_R, k_P, and k_F are proportional constants and F is the lift force generated. If we assume common proportionality, that is $k = k_Y = k_R = k_P = k_F$, and that $F = \sqrt{\sum_{i=0}^{4} T_i}$ then we obtain:

$$\begin{bmatrix} \dot{\psi} \\ \dot{\phi} \\ \dot{\theta} \\ F \end{bmatrix} = \begin{bmatrix} k & -k & -k & k \\ k & k & -k & -k \\ k & -k & k & -k \\ k & k & k & k \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{9}$$

The inputs from the controller of the operator give the desired T, $\dot{\psi}$, $\dot{\phi}$ and $\dot{\theta}$. Therefore to get the motor speeds, we calculate:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = K^{-1} \begin{bmatrix} \dot{\psi} \\ \dot{\phi} \\ \dot{\theta} \\ F \end{bmatrix} \tag{10}$$

Where K^-1 is the inverse of the proportional gain matrix in equation 9 [23].

To obtain the desired thrust, we obtained available data from Cobra CM-2217-20 $Kv = 950$ from [24]. This data allowed us to relate throttle to thrust and then propeller speeds to thrust. The relationships were formalised through polynomial regression.

4.2 Control

To fully simulate how a drone would perform in a real environment it is not suitable to adjust anything other than the torque and moments created by the motors. This means that a control system has to be used rather than directly specifying the UAV's position and attitude in the environment. Fig. 4 shows the control system as it stands; positional and attitude control will be explained in the following subsections.

4.2.1 Attitude

The attitude of a UAV is the difference in the rotation of the UAV's reference frame relative to the world reference frame; Fig. 5 shows both of these reference frames. To control the attitude of the UAV a proportional and derivative (PD) control system was implemented. A PD control was chosen because it is guaranteed to be exponentially stable for most rotations [25]. The PD control system is given by:

$$R = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{11}$$

$$E = R_d - R_a \tag{12}$$

$$\dot{R}_c = -(K_P E + K_D \frac{\delta E}{\delta t}) \tag{13}$$

Where R_d is the desired attitude given by the user, R_a is the actual UAV attitude, $\dot{R}_c$ is the required rotational velocities to make the actual and desired attitudes the same. K_P and K_D are the proportional and derivative constants, and t is time.

To simulate the UAV that is anticipated to go into the mine, the rotation for roll and pitch was limited to 30°, which is equivalent to stabilised mode, with the option to go into acrobatic mode to perform advanced aerial acrobatics such as flips and barrel rolls, if necessary.

4.2.2 Position

An external control loop was added to create the ability for the quadrotor to move to a desired position. The positional controller acts as an external loop to the attitude control. The user or cognitive model feeds the desired position to the positional controller which outputs the desired attitude that is required to achieve this position. This controller is a proportional, integral, and derivative (PID) controller and this was given by:

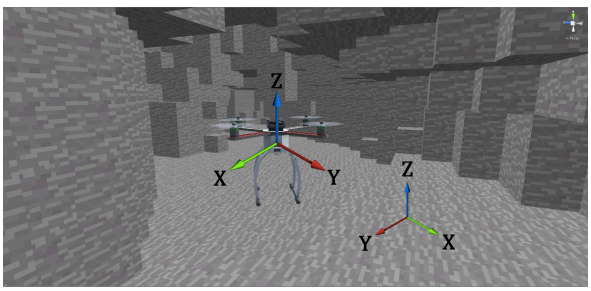

Fig. 5: UAV and World Reference

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{14}$$

$$\epsilon = P_d - P_a \tag{15}$$

$$P_c = -(K_P P + K_D \frac{\delta P}{\delta t} + K_I \int_0^t P dt) \tag{16}$$

$$\dot{R}_d = \begin{bmatrix} -P_c(z)\sin(\psi) + P_c(x)\cos(\psi) \\ P_c(x)\sin(\psi) + P_c(z)\cos(\psi) \\ R_d(y) \end{bmatrix} \tag{17}$$

Where P_a is the actual position of the UAV, P_d is the desired position of the UAV, and Pc is the initial controller output. Pc gives the desired roll and pitch rotations and throttle required to correct the x, y, and z offset. $\dot{R}_d$ is the rotations desired to be put into the attitude controller. The positional controller does not affect the yaw. The yaw is used to rotate the required roll and pitch in the UAV's reference frame relative to the world reference frame. Fig. 5 shows how the UAV's and world reference frame can differ within the voxelised mine model.

4.3 Communication

Prior research into interfacing ACT-R with Unity saw messaging passing over TCP through JSON associative arrays [13], [6].

REACT-R communicates to the robot embodiment through a UDP connection. The reasoning behind using the UDP protocol is that it is faster than TCP, and error checking can be a hindrance to the real time flow of information. There is a constant stream of packets being sent to the cognitive model and for it to react appropriately: the most recent piece of information is crucial. TCP will repeat packet requests when transmission is lost, which can create a backlog. This effectively creates a time lag between what is going on in the real world and what the cognitive model is receiving.

For example, in relation to visual perception, sometimes we fail to see or become aware of an object when it first comes into our field of view. This could be seen as some failure of our visual processor to identify the object. However, we are not concerned with where the object was when it first came into the field of view, but where it is now that we are aware of it and can make a decision about how to react to it. Similarly, current information should take precedence within the decision making process, and as a constant feed of packets is being sent from all sensors the most recent packet is the most relevant.

The cognitive model's REACT module has two buffers, a sensor and an actuator buffer, the sensor buffer receives information from the robot embodiment and the actuator buffer relays information to the robot embodiment.

The UAV simulator uses sonar sensors to understand relative distances from it to collidable objects. These distances are sent to the cognitive model for evaluation along with a user specified status packet that determines the level of autonomy. The packet that is given to the model in this specific scenario is six bytes: one for status and five for sonar sensors. The model will then make a decision about what direction or combination of directions it should move in. The directions are up, down, left, right, forwards, and backwards. The packet that it returns to the simulation is: status and which directions it wants to move in. This also takes into account the goal destination if it is in full autonomy mode.

A ROS node is used to facilitate this communication so that it may integrate easily with other robotics platforms already using ROS. The ROS node can also be bypassed and directed by UDP connection between REACT-R and any platform to be used.

5. Discussion

5.1 Reusable Module

One of the overarching objectives within the scope of the research was to create an interchangeable ACT-R module which is capable of interfacing with multiple external applications. The same rule sets of productions used by ACT-R in one scenario of embodiment can be interfaced with a simulated embodiment, or extended to physical embodiment, given the sensory input is modelled in all embodiments to reflect the same sensory phenomena. The REACT-R module programmed for this purpose requires an IP address and port over which information can be sent. When an IP address and computer port are specified, ACT-R will send and receive packets over the stream regardless of whether the embodiment is virtual or physical, thus allowing for modeling scenarios, flight tests and simulated missions to ensure the productions are accurate, succinct and easily testable prior to testing ACT-R productions on a physical UAV. Through the use of the REACT-R module, sensory inputs and reactive outputs from ACT-R are easily modifiable, as are the port and IP address to which packets are sent and received.

5.2 External interface architecture

5.3 Future Development

Future scope within the development of the REACT-R module will focus on the elaboration of productions within the UAV model and furthering the complexity of interfacing with the physical world through ROS.

5.3.1 Production Elaboration

Within the context of autonomous UAV surveying, productions and rule sets will be further established to provided the cognitive architecture with a greater sense of autonomy. Currently the cognitive A.I has programmed responses to allow for positional control, obstacle detection and obstacle avoidance. As time and research progresses into interfacing the physical UAV with Unity, the complexity of productions given the sensory inputs into the cognitive architecture can expand. These would include productions to allow for

optimal path recognition, flight surveying with additional sensory inputs, mapping of terrain and a further refinement of responsive rule sets to sensory inputs.

5.3.2 Long Term Memory

As ACT-R's declarative memory chunks are fired in productions to complete tasks, the success rate of productions is updated in such a way that frequently coupled productions are automatically merged. The automatic merging of processes allows ACT-R to optimise the decision making process by minimising cost of time for decision calculation. While production merging can aid ACT-R in the process of self optimisation, the drawback of its current implementation is that when the program is terminated all self-created productions and optimisations are lost. Every time ACT-R is restarted, the declaration merging for optimisation needs to rediscovered within the uninterrupted software run time. Providing ACT-R with a long term memory module which saves self-established productions alongside human-programmed productions would ensure that knowledge attained from past experience within an embodiment could aid in the current and future decision making of the cognitive architecture. Future research will see ACT-R retaining self created productions through a long term memory buffer to utilise previously stored experience which would be added to the production rule set on start up and saved when ACT-R is shut down. From a research perspective it will be fascinating to see how ACT-R's self-created rule sets vary from those programmed by humans over an extended period of time.

5.3.3 ROS Optimsation and Elaboration

Within the current state of implementation, ACT-R successfully operated as a node within the Robot Operating System suite, both sending and receiving packets to a physical UAV operating on a ROS system. Future research will include ACT-R performing the decision making process as an autonomous UAV within the physical world. As a cognitive architecture, ACT-R could perform the high level decision making for ROS, selecting which algorithms would be best to apply depending on context-based scenarios. Flight productions and rule sets would first be tested and refined within the virtual environment on the virtual UAV embodiment before being applied to the physical drone embodiment. The REACT-R module provides a streamlined method of passing messages between any client-server based architecture with the hope of modeling behaviour in a simulation before applying the cognitive architecture to a physical embodiment that can meet real world challenges.

6. Conclusion

We have developed a physics simulation of a UAV that validates the use of REACT-R as a simple and easily customisable platform for use on a robotic platform. Using ROS provides ease of integration with currently standing projects.

We have also demonstrated a simulated UAV embodiment of REACT-R that is capable of navigating to a position and avoiding obstacles, as well as assisting pilots in flight by providing hover control and obstacle avoidance. It is anticipated that AI will not replace human operators but rather augment their experience, and we have demonstrated the facilitation of this.

Since REACT-R is anticipated to be used in conjunction with other robots in a multi-agent system, as well as humans, using the core ACT-R cognitive architecture is justified because it is designed with the intent of modeling human behaviour to give robot embodiments better understanding of their human counterparts or team mates.

The elaboration of future work was done in the hope that the logical expansion of the model outlined in this paper can be realised so that we may have harmonious co-operation between human and robot workers in the future.

References

[1] M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009.

[2] E. Tira-thompson and A. D. S. Touretzky, "Tekkotsu : A Rapid Development Framework for Robotics," *Robotics*, 2004.

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

[4] G. Indiveri and S.-c. Liu, "Memory and Information Processing in Neuromorphic Systems," *Proceedings of the IEEE*, no. 612058, 2015.

[5] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures : Research issues and challenges," *Cognitive Systems Research*, vol. 10, no. 2, pp. 141–160, 2009.

[6] R. M. Hope, M. J. Schoelles, and W. D. Gray, "Connecting ACT-R to the World with JSON over TCP," *Proceedings of the 12th International Conference on Cognitive Modeling*, no. rm 108, pp. 354–355, 2013.

[7] W. G. Kennedy, M. D. Bugajska, M. Marge, W. Adams, B. R. Fransen, D. Perzanowski, A. C. Schultz, and J. G. Trafton, "Spatial Representation and Reasoning for Human-Robot Collaboration," *Architecture*, vol. 22, pp. 1554–1559, 2007.

[8] J. G. Trafton, A. M. Harrison, B. R. Fransen, and M. D. Bugajska, "An embodied model of infant gaze-following," *International Conference of Cognitive Modeling*, no. 2003, 2009.

[9] T. Deutsch, C. Muchitsch, H. Zeilinger, M. Bader, M. Vincze, and R. Lang, "Cognitive decision unit applied to autonomous biped robot NAO," *IEEE International Conference on Industrial Informatics (INDIN)*, pp. 75–80, 2011.

[10] P. Langley, "Intelligent Behavior in Humans and Machines," *Aaai*, 2011.

[11] J.-Y. Puigbo, A. Pumarola, C. Angulo, and R. Tellez, "Using a cognitive architecture for general purpose service robot control," *Connection Science*, vol. 00, no. 00, pp. 1–14, 2008.

[12] F. E. Ritter, D. Van Rooy, R. St. Amant, and K. Simpson, "Providing user models direct access to interfaces: An exploratory study of a simple interface with implications for HRI and HCI," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 36, no. 3, pp. 592–601, 2006.

[13] P. R. Smart, T. Scutt, K. Sycara, and N. R. Shadbolt, "Integrating ACT-R Cognitive Models with the Unity Game Engine," no. 2000, 2012.

[14] P. Smart, K. Sycara, and C. Lebiere, "Cognitive Architectures and Virtual Worlds : Integrating ACT-R with the XNA Framework."

[15] F. Tanaka and S. Matsuzoe, “Children Teach a Care-Receiving Robot to Promote Their Learning: Field Experiments in a Classroom for Vocabulary Learning,” *Journal of Human-Robot Interaction*, vol. 1, no. 1, pp. 78–95, 2012.
[16] C. Wei and K. V. Hindriks, “An agent-based cognitive robot architecture,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7837 LNAI, pp. 54–71, 2013.
[17] R. A. Wilson and L. Foglia, “Embodied Cognition,” 2016.
[18] J. R. Anderson and C. Lebiere, “The Newell Test for a theory of cognition.” *The Behavioral and brain sciences*, vol. 26, no. 5, pp. 587–601; discussion 601–648, 2003.
[19] A. Newell, *Unified Theories Of Cognition*. Harvard University Press, 1990.
[20] J. R. Anderson, “A Simple Theory of Complex Cognition,” pp. 355–365, 1996.
[21] J. Whitehill, “Understanding ACT-R âĂŞ an Outsider ’ s Perspective,” *Mplab.Ucsd.Edu*, pp. 1–12, 1993.
[22] J. R. Anderson, M. Matessa, and C. Lebiere, “ACT-R: A Theory of Higher Level Cognition and its Relation to Visual Attention,” pp. 439–462, 1997.
[23] C. Santoro, “How does a Quadrotor fly? A journey from physics, mathematics, control system and towards a "Controllable Flying Object",” Catania, pp. 1–64, 2014.
[24] L. Miller, “How to use the Multirotor Motor Performance Data Charts,” Innov8tive Designs, Inc., Vista, Tech. Rep.
[25] R. Mahony, V. Kumar, and P. Corke, “Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.