Using a Physics Engine in ACT-R to Aid Decision Making

Using A Physics Engine and Simulation for Physical Prediction in a Cognitive Architecture

David Pentecost[1], Charlotte Sennersten[2], Robert Ollington[1], Craig A. Lindley[2], Byeong Kang[1]
Information and Communications Technology[1]
Decision Sciences Program[2]
University of Tasmania[1]
CSIRO Data 61[2]
Hobart, Tasmania
[1]Australia
email: davidp12@utas.edu.au
email: charlotte.sennersten@csiro.au
email: robert.ollington@utas.edu.au
email: craig.lindley@csiro.au
email: byeong.kang@utas.edu.au

***Abstract*—Advanced Cognitive Technologies can use cognitive architectures as a basis for higher level reasoning in Artificial Intelligence (AI). Adaptive Control of Thought – Rational (ACT-R) is one such cognitive architecture that attempts to simulate aspects of human thought and reasoning. The research reported in this paper has developed an enhancement to Adaptive Control of Thought – Rational (ACT-R) that will allow greater understanding of the environment the AI is situated in. Former research has shown that humans perform simple mental simulations to predict the outcomes of events when faced with complex physical problems. The method has been demonstrated by application in an autonomous squash player for which, predictive version of ACT-R achieves significantly improved performance compared with the non-predictive version.**

Keywords- Cognitive Architectures; ACT-R; 3D Simulation.

I. INTRODUCTION

This paper is an extended version of the paper "Predictive ACT-R (PACT-R)" published in "COGNITIVE 2016 : The Eighth International Conference on Advanced Cognitive Technologies and Applications" [1].

A challenge to robots in the real world is dealing with complex fast changing environments with fast paced decisions where there is little time for deliberation. This means that we need systems that can rapidly perceive, act and reason.

How could a cognitive robot – that is, a robot endowed with deliberative problem-solving – track and interact with fast moving objects in a complex environment? How could a robot interact or perform actions in a dynamic situation?

What do you do if you are asked to catch a ball that has been thrown in the air? You make a quick estimate of its trajectory, predict where you need to be to intercept it, and then move to that location. What about if it is going to bounce off a surface? Although there is now a little uncertainty, if you do not know the properties of the ball and surface, it is, nevertheless, not much more difficult to make a good enough prediction and correct for any errors after the bounce. What about if the ball bounces several times before you reach it? Now, you are more likely to start looking at the likely chain of events that will occur to predict the outcomes.

Artificial Intelligence (AI) in robotics commonly uses either an algorithmic approach, that is, a custom solution to a specific problem [2], or subsumption-like architectures that react to the world as it is perceived [3]. Reinforcement learning offers another approach, which allows agents to solve problems without expert supervision by interacting with their surrounding environment to learn cause and effect relationships. Reinforcement learning allows a robot to discover a behaviour through a trial and error approach without requiring explicit instructions for the solution [4].

The algorithmic approach is effective for well-understood problems with little variation, but it is not so good at responding to the unexpected. Subsumption follows a 'stimulus and response' model. It is good at dealing with immediate problems, like avoiding obstacles, but can be lacking when it comes to a multi-stage mission that may require evaluation and decision-making over several alternative sequences of actions. Cognitive architectures have been proposed as an alternative that could be more suitable for accomplishing missions that require sequences of decisions, rather than more purely reactive associations between sensor inputs and motor outputs.

The American Physiological Association defines cognition as, "Processes of knowing, including attending, remembering, and reasoning; also the content of the processes, such as concepts and memories." Cognitive architectures are based on theories of how the human mind reasons to solve problems. They are used to create AIs based on, or inspired by, human cognitive processes that work through problems in a systematic way [5]. They are based on a Computational Theory of Mind, which holds that the mind works like a computer, using logic and symbolic information to work through and solve problems. Symbolic information is, in a programming context, a textual/verbal approach to

representing knowledge in a way that is abstracted from sensory data, since the relationships between words and their referents are conventional. This abstraction supports potentially complex symbolic reasoning processes, but omits much detailed information about objects and phenomena that the symbols refer to in a given context.

Hence cognitive architectures, like other approaches to AI, have their own limitations. For example, they are similar to expert systems [6][7][8] in using facts and production rules that require a human expert to create. They are strong at symbolic reasoning with logic, but the ontological status of symbols within human cognition is unclear [9], and the biological foundations of human cognition are very different from the nature of expert systems and formal logics [10]. Expert systems and formal logics are technologies, i.e., inventions of human cognition, rather than its basis. They may, nevertheless, be useful and even powerful representations of some human capabilities that are based upon much lower level biological mechanisms.

An aspect of human cognition that is not captured in most cognitive architectures is *simulation*. Imagination, and the use of imagined visualisations, constitutes a conscious result of simulation within human cognition. An example of the use of simulation in an artificial cognitive system is the Intuitive Physics Engine (IPE), which uses simulation to understand scenes [7]. IPE uses a fast approximate simulation to make a prediction of the outcome of a physical event or action, like the toppling of a stack of blocks.

In synthesizing a world, simulation provides a cognitive system with the richness of a sensed world, with far more detail than that, which can easily be captured in higher level symbolic world descriptions alone and at a potentially much faster speed than the perception of external processes bound to the physical process speed. Simulating a 3D world and aspects of its physics involves using mathematical models of world structure, kinematics, dynamics and object interactions in, which complex behaviours can be synthesized from a relatively small set of structural and physical equations. The quantisation of space and time in a simulation can be represented, e.g., to double floating point precision, resulting in an extremely large space of possible simulated world states and histories. The level of abstraction involved in declarative or symbolic representations is usually much higher than a simulated world state description, since it is expressed at a level suitable to verbalised decision processes, meaning that many simulation states can be compatible with a single declarative representation. That is, a declarative statement can provide a succinct and abstracted representation of a large set of world state denotations. For example, the first order predicate '*is_above*(A,B)' can apply to any object in a simulation that is above another object. But to represent all of those possible individual denotations (every possible situation and variation of positions in, which one object is above another) declaratively would be practically impossible. The declarative level of decision processing can be linked to the simulation state, e.g., via spatiotemporal operators linked to the simulation structure, such as testing for the relative 3D positions and sizes of objects A and B as a basis for assigning a truth value to the statement '*is_above*(A,B)'. Hence there is a useful balance between what can be represented and reasoned about most effectively using declarative representations, and the large number of potential states having small differences representable by a simulation. These are complementary modelling methods. To explore these concepts, this paper describes an experiment designed and implemented to further test the theory that simulation is a powerful component of cognition. The motivating research question was: "How can simulation and prediction improve decision quality in a cognitive architecture?" In the experiment designed to address this question, a predictive module was added to a cognitive architecture, and the performance of the predictive and non-predictive versions of the architecture were tested for controlling automated players of a virtual game. The predictive module used a 3D physics simulation engine to model the environment of an embodied AI, so that it could function in a dynamic situation without explicit coding of decision rules for all possible interactions in the environment. The simulation engine mathematically models interactions with the environment so that the cognitive module can handle physical events and actions with a reduced and simplified rule set.

An existing cognitive architecture, Adaptive Control of Thought – Rational ((ACT-R) [12][13][14]), was chosen for the research and extended with a novel predictive module. Two virtual robots were implemented to play a competitive game of squash (Figure 1). Squash is a racket and ball sport played in an enclosed room between two players. It was chosen because it provides both a physics challenge (tracking and hitting the ball), and a cognitive challenge (playing a good tactical game to out-manoeuver an opponent).

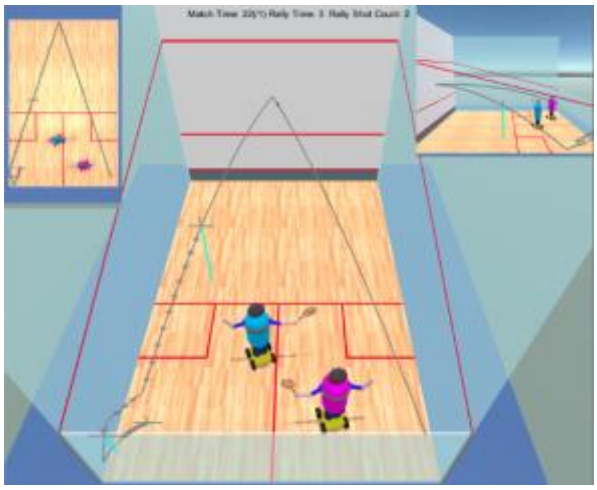

Figure 1. Squash Simulation showing AI controlled players. Predicited ball path is shown as grey track. Vertical lines are where the AI player can intercept the ball.

Section II of this paper gives some background to cognitive and non-cognitive architectures. In Section III a description of the research undertaken and methodology used is given. Section IV describes the AI modelling and how prediction was incorporated. Section V discusses the results obtained.

II. AI ARCHITECTURES FOR ROBOTICS

Cognitive architectures are based on theories of how the human mind reasons to solve problems. These are AI systems based on human cognitive processes that work through problems in a systematic way [5]. They are based on the Computational Theory of Mind [15], that proposes that the mind works like a computer running a program, using logic and symbolic information to work through and solve problems.

The cognitivist approach is a top down approach that follows a rule-based manipulation of symbols, and uses patterns of symbols, as designed by humans, to represent the world [16]. A key characteristic is that the mapping of perceived objects to their associated symbols is either defined by humans, or learned in a way that can be viewed and interpreted by humans. Decisions about, which actions to perform are derived by processing of the internal symbolic representations of the world. In contrast, the connectionist approach is a bottom up approach that proposes that intelligence is an emergent property of connected cells/nodes and networks; "Neurons wire together if they fire together" [17]. Within the cognitivist approach we will here focus on the cognitive architecture called ACT-R. ACT-R is a cognitive theory [14] that has been simulated in software as a production rule based, symbolic data model that includes elements of the connectionist view as sub symbolic data. It is usually referred to as a hybrid architecture.

Using cognitive architectures for robotics is described by Laird et al. in the adaptation of Soar (a similar cognitive architecture to ACT-R) for robot control [18]. For the robotic control task, Soar was extended to include mental imagery, episodic and semantic memory, reinforcement learning, and continuous model learning. Soar and ACT-R share features including procedural memory encoded as production rules, and semantic memory implemented as declarative associations. A number of architectures similar to Soar and ACT-R are reviewed in [19].

Most current operational robots do not use cognitive architectures. Instead, traditional robotic research and control has focused on software that solves problems having well formulated solutions; this can be referred to as the *algorithmic approach* [2]. These systems are particularly suited to well-defined tasks and domains. However, there is a need for higher level cognitive abilities to deal with less well defined problems. Unknowns are a challenge for robots where the problem is not fully specified. The strength of human cognition is the ability to solve unknowns in new or changing contexts. It is in these situations that cognitive architectures, which are based on theories of human cognition, offer potential capabilities for solving problems in uncertain situations.

The subsumption architecture is another alternative to cognitive architectures for robot control. The subsumption architecture approaches intelligence from a different perspective. Rather than rules that lay out a series of steps to accomplish a task, it uses a very sparse rule set that responds to sensor values to generate control outputs [20][21][22]. Brooks describes subsumption as a layered finite state machine where low-level functions, like "avoid obstacles", are subsumed into higher-level functions, like "wander" and "explore". Each successive layer gives increasing levels of competence. Lower levels pre-empt the higher levels, such that a robot can explore, but will avoid obstacles when necessary.

Key aspects of subsumption are: that it contains no high level declarative representations of knowledge; no declarative symbolic processing; no expert systems or rule matching; and it does not contain a problem-solving or learning module [2]. It responds to the world by reacting directly to sensor inputs, in order to generate corresponding control outputs.

Subsumption is based on the concept that the environment stands for itself, i.e., the architecture reacts directly to environmental features, without a mediating representation. It is a functional architecture without being, or using, a declarative model of the external world. This approach can be complementary to the cognitive architecture approach. The concept behind subsumption is a stimulus response model that is ideal for modelling low level instinctive behaviours, which can easily be incorporated into other cognitive architectures.

A robotic AI might be created completely within a single cognitive architecture, using rules that control every aspect of the decision-making process. But monolithic architectures are not always ideal for every style of decision-making. Society of Mind [24] is a theory that argues against the idea of a single unified architecture or solution that can account for intelligent behaviour. It proposes that intelligence arises from many simple specialised functions working together [23][24]. Hence Society of Mind theory argues for a modular approach to implementing intelligence. Implementing simulation as an extension to a cognitive architecture, but using an external 3D engine to model the environment, follows this concept.

ACT-R is a hybrid cognitive architecture consisting of both symbolic and sub-symbolic components [25]. It is a goal-orientated architecture. The symbolic data consists of facts and production rules. The sub-symbolic data is metadata about facts and production rules that influences, which facts are recalled and, which production rules are chosen to fire when multiple facts and rules fit the current situation. The sub-symbolic data can be seen as a hidden layer controlling the selection of rules not unlike the weights in the hidden layers of a neural network. This is a hybrid of symbolic and connectionist models of cognition [14].

ACT-R consists of multiple cognitive modules that implement either internal or external functionality. For example, a memory module would be an internal cognitive system that could hold declarative knowledge (facts and figures). A vision module would be an external system giving a representation of the external world to the architecture. Central to the Cognitive Architecture is the Procedural module for selection of rules, Figure 2.

The Procedural Module is a pattern matching system, it looks for combinations of values held in buffers. The buffers represent the current state of the cognitive modules that form the architecture. The contents of a buffer are a set of key name and value pairs. The combination of the content of all the buffers represents the architecture's current state. The patterns that the procedural module looks for are specified in

production rules. A production rule is an if-then statement, i.e., that *if* some condition (pattern) is satisfied *then* an action is fired. The *if* part of a production rule looks for particular values in one or more cognitive module buffers; for example, it might specify that the visual module buffer contains data that a blue box is visible in the upper right corner of the view while the goal module is indicating that it should be looking for an object. If these conditions are met, then that rule is a potential candidate to be fired.

At its simplest the Procedural Module will randomly choose a rule to fire from amongst those that match the current conditions. However, the sub symbolic system can influence rule selection. Sub symbolic data can be seen as a weighting system for rule selection and these weights are either fixed or can be learnt through reinforcement learning.

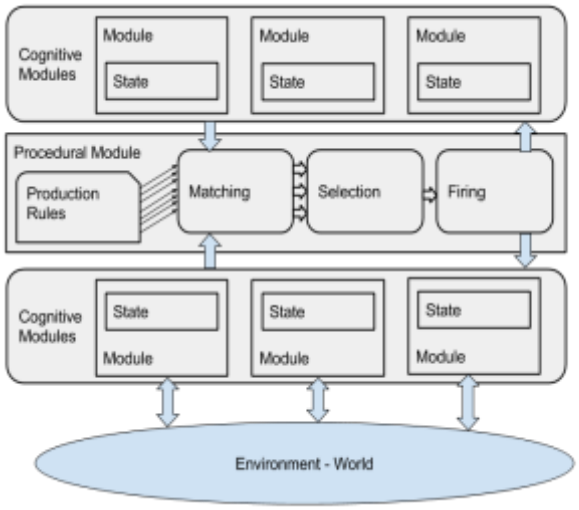

Figure 2. Basic structure of an ACT-R Cognitive Architecture

III. METHODOLOGY

This section describes the research design, and the implementation of the prediction and simulation extensions to ACT-R to constitute the Predictive-ACT-R (PACT-R) architecture.

A. Research Design

The research consisted of developing and implementing three elements: 1) the PACT-R cognitive module that implemented the simulation-based prediction system; 2) a virtual environment for testing; and 3) three AI models to test the system. The models are so called behavioural and functional scripts in the language of ACT-R. An ACT-R cognitive module was developed that mapped a symbolic representation of a simulated environment into the ACT-R framework. This module gave the required PACT-R functionality for interpreting and acting within the environment, as well as providing simple predictive capabilities using simulation.

The use of prediction and simulation in PACT-R was evaluated by comparing the performance of three AI models that each implemented different levels of prediction. The aim was to compare not only their performance, but also how easy/simple it was to model and use a predictive AI.

B. Implementation

The first of the three elements mentioned above, PACT-R cognitive module that implemented the simulation-based prediction system, gave the testing models (scripts) access to predictions about physical events, as well as a mechanism to take actions. The module was implemented in the Lisp programming language that ACT-R is implemented in, and added to the ACT-R code base.

Two predictive elements were added. The predictive module always provided a prediction of the ball's flight path for the purpose of intercepting and hitting the ball. A further predictive element was added that allowed the AI model to evaluate its own possible actions within a simulation to determine the likely outcome of those actions. Essentially, the model could ask very simple "what if?" questions about how its own actions might play out in the future.

The second element was a simulation of the game of squash implemented in the Unity™ game engine. The physics simulation element of PACT-R was also implemented in Unity™ to simplify implementation. The Lisp and Unity elements of PACT-R were connected using a UDP connection.

Squash is a racquet sport played in a closed room between two players. The ball is free to bounce around the walls, and a player is free to hit the ball against any wall as long as it reaches the front wall before a second bounce on the floor. The opponent also must reach the ball and play a shot before the second bounce. This requires fast paced decisions within a one to two second period. To play squash our cognitive model must make decisions within a frequent turn based action cycle.

The game of squash has been described as physical chess, since it is both physically demanding and highly tactical. The physical challenge is a result of the continuous explosive acceleration needed to react to, and retrieve, an opponent's shot.

The tactical element of the game plays out in the shot selection and how this can be used to apply pressure to the opponent. When deciding when and where to hit the ball the player is faced with many choices. Do they take the ball early before it reaches a wall? Do they wait and give themselves more time to play a better shot, knowing that this also gives their opponent more time to move to a stronger court position? Is a shot to the front of the court the right shot? It puts the opponent under more physical pressure, but if they reach it, with a bit of time to spare, it opens up a lot of attacking shots.

Squash is also a game of angles, much like a real-time game of snooker. Judging and playing the angles is an important part of the game. Using squash as the test scenario provides the known rule set for the game and existing tactical knowledge for implementing the AI models.

One important aspect of squash, that was not included in the implementation, was player fatigue. Players get tired and shot selections can be made to tire an opponent rather than simply winning the current rally. A tired player will move

more slowly and tend to make weaker shots. This factor was excluded from the experimentation to reduce the number variables to the minimum needed.

In the following discussion, the games engine that hosts the squash simulation will be clearly distinguished from the physics engine that makes predictions. Due to time constraints, these shared a common code base, although it would have been more correct if the prediction physics used a rapid approximation of the environment. The game engine physics allows a deterministic perfect prediction that would be unobtainable in a real-world scenario. In the real world, there is uncertainty due to inaccuracies in both perception and motion. To compensate for this, the prediction physics were degraded by introducing randomness into the motion (2% to speed and direction) and by using a very low resolution interpretation of shot outcomes (only 3 interpretations of outcome; favourable, unfavourable and neutral).

The final element modelled squash-playing AIs as test scripts. Three evaluation test scripts were developed for testing and cross-comparison (see Figures 6 and 8 for two of these).

The following sections describe the high-level design of PACT-R and continues with specific implementation details.

1) Using Simulation and Prediction within a Cognitive Architecture

The research investigated the use of a physics engine to provide prediction for a cognitive architecture. The concept requires a physics engine that can model and simulate the environment of a robot controlled by a cognitive AI. The simulation provides a symbolic representation of the environment to a cognitive architecture. This gives the cognitive model (the production rules) the information it needs to understand and act within its environment.

One way of using this information could have been to explicitly encode rules that check for certain conditions, for example, whether an object is in a certain position, or is moving in a particular direction; or for the relationships between objects in the environment, for example, whether an object is to the left of another object [26][27]. From this, the rules can encode appropriate actions for the robot to take. This approach was the basis of the heuristic test model describe below.

We researched and explored an alternative approach that broke down the decision-making process into two phases. The first phase determined a set of possible actions that fit the current situation, implemented as a script in ACT-R. The actions, such as a low-hard-straight shot or a high-cross-court-lob, were passed sequentially to the physics simulation to predict their likely outcome. The second phase then picked the best of these as the action to be taken. Figure 3 shows a high-level diagram of this approach. The action with the best predicted outcome is the one that is fired.

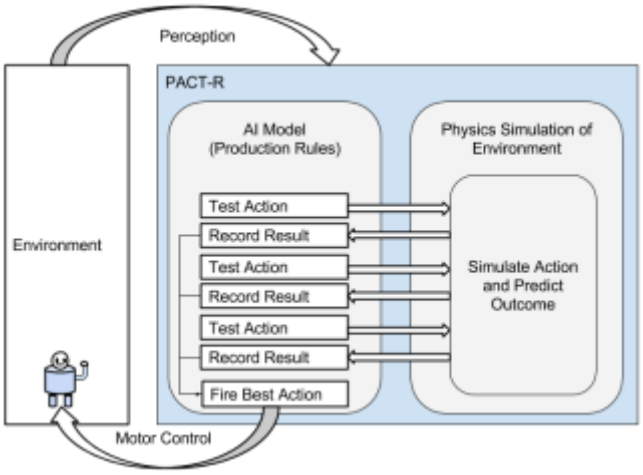

Figure 3. Overview of PACT-R. The simulated world is modelled in a physics engine. The cognitive model (a set of production rules) tests possible actions in the physics engine to determine how they play out before selecting an action to execute.

Below we proceed to break this overview down to describe the low-level implementation within the ACT-R cognitive environment.

2) PACT-R Module Implementation in ACT-R

The prediction system is implemented as an ACT-R module that both controls a robot and does a simulation of the robot's environment, for interpreting what is happening in that environment. The module is, logically, a single system, but in the implementation, it is broken into two functional parts, one residing in the ACT-R framework, and the other inside the Unity™ game engine, which includes a physics engine and also hosts the virtual world the robots exist in (Figure 4).

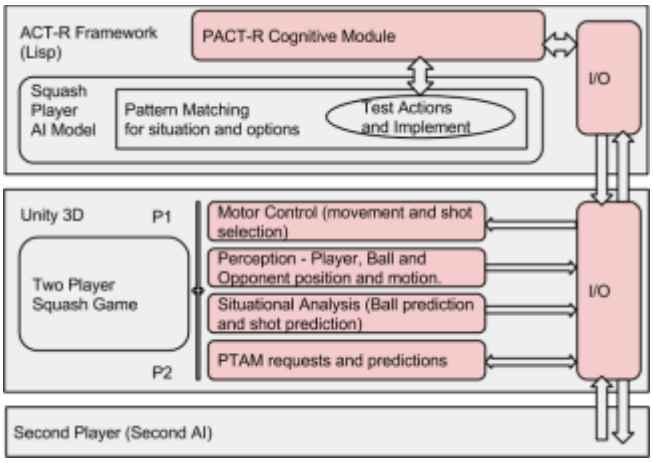

Figure 4. PACT-R (in red) within ACT-R and Unity. PACT-R adds a cognitive module to ACT-R that allows testing of actions as well as an interface to the external world (simulated in Unity)

The ACT-R component of the system maintains the current simulation and prediction state for use by the AI models, while the Unity™ component of the system contains a customised physics engine that can simulate both the squash ball's path, and the outcome of shots played by the robot. The

two components of the module connect via a Universal Datagram Protocol (UDP) link, a standard part of the Internet Protocol (IP).

For PACT-R, the cognitive module represents implicit knowledge (of the sort that a squash player learns over many years) and the simulation. Implicit knowledge encodes the 'how' of movement and ball striking, while the simulation provides a basis for deciding 'what' and 'where'.

The PACT-R enhancement to ACT-R is implemented as an additional ACT-R cognitive module, Figure 4, that provides two buffers, one that commands are sent to, and the other that gives the model access to a simplified view of the environment (situation). The module communicates with the simulation engine, in Unity3d, to request predictions through the command buffer and to receive the predicted outcomes through the situation buffer.

IV. AI Modelling and Prediction

This section presents the outline of the AI models at a conceptual level, rather than dealing with the details of modelling them in ACT-R. Then, the implementation of the prediction module in ACT-R is presented, together with its interactions with the AI models, followed by a description of the evaluation and analysis framework for these models.

A. Prediction Models

The simulated task, playing squash, that the AI must perform is dynamic; the ball is in continuous motion and can follow complex paths as it interacts with the walls and floor. Likewise, the AI's robotic avatar is moving, as is the opponent.

ACT-R is designed to look for and respond to patterns in information in its buffers. The buffers hold information representing both the external world and the AI model's internal state. ACT-R can work with values and do simple comparisons, but doing complex calculations and mathematical evaluations is not its forte (although it is possible to call Lisp functions if required). It is the role of the modules, particularly those that deal with the external world, to abstract the situation into a simple symbolic representation that the AI model can reason about, by searching for patterns and relationships.

For a complex dynamic situation, this may present a problem, since an AI model requires deliberation (i.e., "thinking") time. That is, it needs time to recognise a pattern and fire a production for the situation the pattern represents. To solve a problem, it may have to fire a sequence of productions. For a dynamic situation, by the time a pattern has been recognised and acted upon, the situation may have already changed to something different.

The simulation-based module described here abstracts away the details of the environment into a simple set of relationships and events representing the elements in the scene. This abstraction is highly domain specific; in the implemented PACT-R, the abstraction focuses on the specifics of the game of squash.

For squash, PACT-R identifies three actors: *self*, *opponent* and *ball*. The module provides the AI model with information about the approximate locations of these actors within the squash court and information about what is happening, is about to happen, or what might happen. Real coordinates and vectors of motion are absent from the information. The aim was to focus on the tactical evaluation so the spatial relationships were evaluated in the PACT-R module and presented to the model in a very simple form.

For this research, a baseline capability of the prediction module included a prediction about the immediate known ball flight path that the AI model could use to intercept the ball, at an appropriate court position, to play a shot. This prediction was made following the opponent's shot when the ball's position and velocity could be determined. The ball's path was simulated in the physics engine, which tracked where the ball would travel as it bounced against the walls and floor. The path was calculated until it was determined that the ball would have bounced on the floor for a second time. This projected ball path was then used in the prediction module to determine locations where the player could intercept and hit the ball, based on their own movement ability.

The intercept positions were placed in the prediction module buffer used by the AI model, which allowed the models to intercept the ball without any further processing. The intercept position could have been under AI control, but this would have introduced more complexity to the modelling and introduced more independent variables to the test, making it difficult to determine cause and effect. For this reason, AI control and reasoning was limited only to the shot selection strategy.

To know where the ball and the player were within the squash court, the squash court was broken into strategic zones and all positions were given as zone numbers. The squash strategy implemented in the models was also based on zones, with a limited selection of shots available for each zone. The AI models selected a shot from those available in the zone where the ball was intercepted. The zones and shots are based on squash training drills commonly used to teach players basic strategy.

B. Evaluation and Analysis

Three AI models were developed and evaluated. The first model was a *basic random shot selection model* that functioned as the base line to determine whether shot selection by the other models was better than random chance.

The second model was a *heuristic model* that had an explicit shot selection rule-set derived from the human developer's experience of playing squash. This model's purpose was to provide an alternative method to the prediction model.

The third model used the *predictive* features of PACT-R to test shots for their likely outcome.

To evaluate the performance of the three models, a large amount of automatic data gathering and logging was conducted from the virtual environment. This data provided both comparative performance of the models, and an insight into how they won or lost.

For each test session the only variables were the shot selection strategies of the two competing AI models. Test sessions consisted of two AI models (out of three) loaded into the ACT-R environment, playing against each other over a

series of rallies. Squash starts with a serve from one player to another. The two players then alternate shots until one is unable to retrieve or return the shot, and therefore loses. In squash this is called a rally. Data recorded included shot selection and state during the rally, and the final results of each rally. This was repeated for a fixed time (from three to eight hours) to generate a large sample set of data.

For a test run (a rally), the serve was alternated so there was no bias or advantage to either model. Player 1 always started on the forehand side (right), and player 2 on the backhand. The players were ambidextrous with no advantage to either side (unlike human squash players). The two robotic players were identical in their capabilities.

V. Results and Discussion

The three models discussed here all follow the same base strategy. They must choose from three or four shots available for the zone where the ball is to be hit. The basic model did not use any additional logic to choose a shot. The other two models tried to choose a shot that would force the opponent to have to travel the furthest to reach the ball in order to play their next shot.

A. Basic Random Shot Selection Model

The first AI model developed was a random shot selection model. This created a setup with three or four equally possible shots for each court zone for ACT-R to choose with its production rules. With no additional conditions in the rules, other than the court zone, a shot would be chosen at random from those available.

This model acted as a baseline control. It was also the only model used during development and balancing of the simulation and physics engine.

B. Heuristic Selection Model

The second model was a heuristic model that used ACT-R production rules that implemented a simple squash strategy that tried to choose shots that would be directed to an area of the court where the opponent was not present. For example, if the opponent was deep in the court (i.e., close to the front wall of the court), it would favour a short shot, and if the opponent was on the forehand side, it would favour a backhand shot. Shot selection rules for each zone were implemented using this simple strategy. In real squash, this approach is a good starting point for any human player.

Figure 5 is a flow chart representation of part of the heuristic model, although it only shows one shot selection choice, rather than the many that were required to model shots for all court zones. It should be noted that for ACT-R production rules, matching and firing does not proceed in a step-by-step fashion like a flow chart. The flow chart representation is used to show the logic, rather than the functioning of the models.

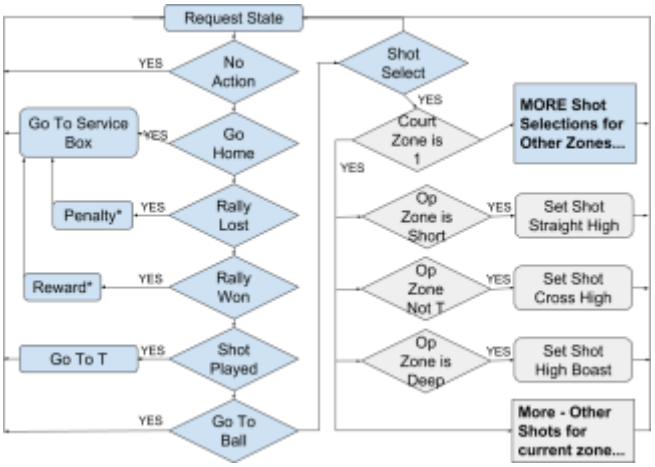

Figure 5. Heuristic AI flow chart. The left hand side shows supporting rules for general play. The right hand side shows one set of rules for shot selection in one of the twelve tactitical zones that the squash court was divided into.

Each diamond and rectangle pair in Figure 5 corresponds to a production rule. The heuristic model consisted of 45 production rules for shot selection, plus another 5 to implement the functionality required for starting and ending a rally, and for returning to a central court position when not returning a shot.

Figure 6 shows just one of the 45 rules required to select a shot for a zone. Each rule defines the conditions under, which it can fire and the action to be taken. In Figure 6 this rule tests where the ball will be intercepted and where the opponent is. This rule says that if the ball is wide (left or right side of court) AND in the mid court (from front to back) AND the opponent is at the front of the court THEN play the ball to the back of the court.

```
(p take-shot-z22-z23-StHi-OpSh
    =goal>
      ISA playing-mode
      state    2       ; play mode
    ?command>
      state free
    =predictive>    ; PACT-R module
      ISA predictive-state ; correct chunk type
      special 5     ; shot selection mode
      > intercept-zone-width 1 ; position wide
      intercept-zone-depth 2 ; position mid
      > op-zone-depth 2  ; op at front of court
    ==>
    +command>
      ISA command-packet
      req-cmd     4       ; Set Shot to play
      :req-param  51      ; Long High Straight
  )
```

Figure 6. Heuristic AI shot example rule (script). This is one rule from the right hand side of Figure 5. The key elements of the rule are in bold.

C. Predictive Selection Model

The third AI model was the predictive model. The random and heuristic models both had access to a prediction of the ball's path that they could use to determine where to go to hit the ball, and consequently, what shots they should be playing, based on where the shot was to be taken.

The predictive model went a step further in predicting the outcome of shots the AI model might take. This was done by allowing the AI model to choose a possible shot before passing that information to the prediction module for simulating and predicting its consequences. The module would simulate how the shot would play out to predict where the opponent would be when the shot was played, and how much difficulty they would have in then retrieving it and playing a counter shot. The prediction was based on the same strategy as the heuristic model, trying to find a shot that was as far from the opponent as possible.

The prediction system has one advantage over the heuristic model: as it is calculating the path of the shot under test, it sometimes found situations that it could not solve for the opponent to intercept with the ball. In essence, it had found winning shots that the opponent could not return. This result was passed back to the AI, which allowed the predictive model to find and choose these occasional winning shots.

Figure 7 shows the prediction model as a flowchart, and a sample rule. Unlike the heuristic model's 45 rules, this model only requires 26 rules for shot selection. Each rule defines a shot to be tested for a particular zone of the court.

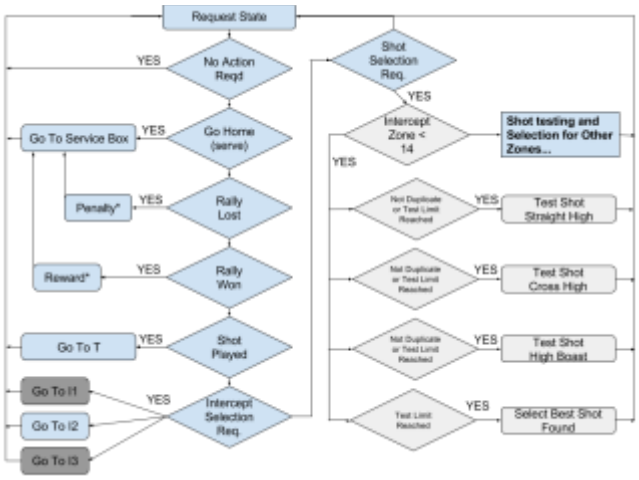

Figure 7. Predictive AI shot selection flow chart. The left side shows supporting rules for general play. The right hand side shows rules for testing shots from one strategic zone.

Figure 8 shows one of the rules for testing shots in the Predictive model. Each such rule sends a shot to be tested to the prediction engine. There were several such rules for each strategic zone. During the shot selection period, while the player was moving to intercept the ball, these would be randomly chosen and fired. The number of predictive tests that could be made was limited to four. The PACT-R cognitive module kept track of the best rule tested so far, and when all applicable rules have been tested another rule would retrieve the best shot and set it as the selected shot.

```
(p take-shot-z22-z23-StHi
    =goal>
      ISA playing-mode
      state    2           ; in play mode
    ?command>
      state free
    =predictive>           ; PACT-R module
      ISA predictive-state ; correct chuck type
      special 5            ; in prediction mode
      < prediction-count 4 ; more testing allowed
      - registered-shot 51 ; not already tested
      > intercept-zone-width 1 ; court pos wide
      intercept-zone-depth 2   ; and mid depth
    ==>
    +command>
      ISA command-packet
      req-cmd     5        ; Test Shot (predict)
      :req-param  51       ; Long High Straight
)
```

Figure 8. Predictive AI shot selection sample rule. This is one of 26 rules that selects a shot to be tested by the physics prediction engine.

The predictive system works by allowing the AI model to test shots that are available to play. This allowed the prediction system to usually come up with the best shot available within the limits of the prediction resolution. Figure 9 shows the progression of the shot testing as the cyan player moves to intercept the shot. The grey track shows the ball's current path in the top right frame. In subsequent frames blue tracks appear that represent possible shots. In the final frame the cyan player has played the best shot found, which is another straight shot down the left hand side (shown in grey again).

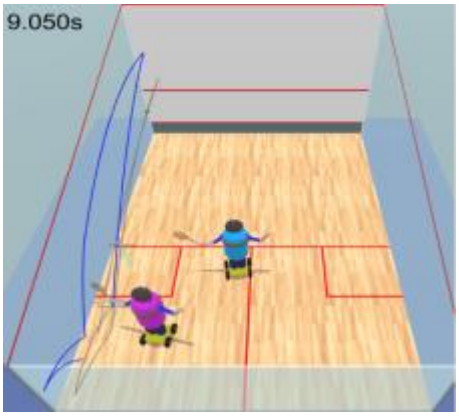

Figure 9. Time lapse of predictive shot selection showing test predictions for cyan robot (center of screen in first frame). Each image shows a new shot being considered (blue tracks). The last frame shows a single track for the final selected shot, in this example this was the first shot tested (second frame).

This sequence of shots takes place over a period 800ms. Figure 10 shows an abbreviated trace of the ACT-R rules firing for the sequence in Figure 9. Prediction tests are 200 ms apart, which is determined by ACT-R's default time for rule selection and firing. The first shot tested scored the highest and is selected as the shot to play in the FINAL-SHOT-SELECTION rule fired at the end of the trace.

```
9.050 PRODUCTION-FIRED TEST-SHOT-Z22-Z23-STHI
   Testing shot 51 0
   better predicted value 2 for 51
9.200 SET-BUFFER-CHUNK SPATIAL SPATIAL-STATE45
9.200 SET-BUFFER-CHUNK SITUATIONAL-STATE45
9.250 PRODUCTION-FIRED TEST-SHOT-Z22-Z23-BODF
   Testing shot 23 1
   predicted value 1 for 23
9.400 SET-BUFFER-CHUNK SPATIAL SPATIAL-STATE46
9.400 SET-BUFFER-CHUNK SITUATIONAL-STATE46
9.450 PRODUCTION-FIRED TEST-SHOT-Z22-Z23-CRHI
   Testing shot 52 2
   predicted value 1 for 52
9.600 SET-BUFFER-CHUNK SPATIAL SPATIAL-STATE47
9.600 SET-BUFFER-CHUNK SITUATIONAL-STATE47
   ...
9.850 PRODUCTION-FIRED FINAL-SHOT-SELECTION
```

Figure 10. ACT-R trace of a test and prediction sequence of rules being fired. This section shows three shots being tested at time 9.050, 9.250 and 9.450 seconds. The rule for retrieving and firing the best tested shot occurs at 9.850, the trace above skips over the period from 9.600 to 9.850 for brevity.

D. Declarative Memory Predictive Model

The research described so far exclusively used production rules to provide both declarative and procedural knowledge. A model based on a declarative memory model was also tested and evaluated after the primary research described in this paper was completed. This model was a reimplementation of the predictive model that replaced explicit rules with declarative knowledge that matched a situation to possible shot selection, as shown in Figure 11. This produces a more concise model since memory declarations are significantly smaller than rules.

This model suffered from performance issues due to the memory recall time imposed by ACT-R. The model could not submit a declarative memory request, retrieve the result and act on it a timely enough fashion to compete against the other models. In ACT-R, memory retrieval has additional time penalties, compared to production rule firing, which aims to emulate human recall performance. Detailed results were not gathered for this method.

```
(chunk-type zone-shot z s)
 (add-dm
   ;; Deep, back of court
   (z12-StHi ISA zone-shot z 12 s 51)
   (z11-StHi ISA zone-shot z 11 s 51)
   (z13-StHi ISA zone-shot z 13 s 51)
   (z12-CrHi ISA zone-shot z 12 s 52)
   (z11-CrHi ISA zone-shot z 11 s 52)
   (z13-CrHi ISA zone-shot z 13 s 52)
   (z12-BoDf ISA zone-shot z 12 s 23)
   (z11-BoDf ISA zone-shot z 11 s 23)
   (z13-BoDf ISA zone-shot z 13 s 23))

 (p find-another-shot
  =retrieval>
    ISA       zone-shot
    s         =sh
    z         =zn
  ==>
  +retrieval>
    ISA       zone-shot
    z         =iz
  - s         =sh
```

```
  +situational>
    ISA command-packet
    req-cmd     5       ; Test Shot
    :req-param  =sh
)
```

Figure 11. Declarative memory and retrieval production rule. The rule has been simplified for illustration. The =retrival> section retrieves the previous memory recall. +retrival> makes the next request. +situational> takes the previous retrieval and passes it to the predictive system for simulation and testing.

E. Performance

The three models that were developed could all play a game of squash since they all encoded some level of squash strategy into shots available in any situation. They differed in the method used to choose the shot from the subset available. Figure 12 shows the player to player performance of all three models. When playing identical models against each other, the results are even, as would be expected. Both heuristic and predictive models won over the basic random selection model, indicating that these models are performing better than chance.

Predicting the outcome of actions before selecting one was better at choosing a winning shot than the heuristic approach. The predictive model won significantly more rallies than the heuristic model. Out of 540 rallies played during a six hour test the predictive model won 312 of them to 228 by the heuristic model. The binomial test p-value for this is 0.0003, showing that this is unlikely to be due to random chance.

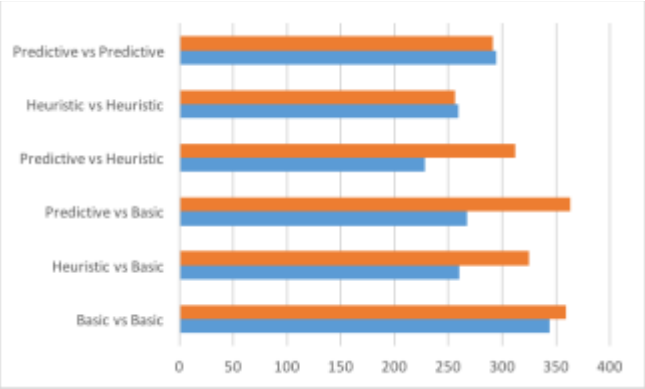

Figure 12. Head to head scores for all models. Each pair of bars is the scores of two tested AI models playing against each other over a six hour period.

When developing the models, there was a clear advantage to the basic and predictive models over the heuristic model in the reduced number of rules required to implement the shot selection strategy. The basic and predictive models required 25 and 26 rules, respectively. The heuristic model required 45 rules to implement a simple shot selection strategy.

The question the original research asked was "How can simulation and prediction improve decision quality in a cognitive architecture?" The answer to this is not straightforward. The results show that, within the limitations of the experiment, a predictive model – with an ability to use simulation to test its own actions to determine and evaluate their possible outcome – held a clear advantage over a model that used heuristics to test relationships between objects in a simulated scenario.

It is not, perhaps, surprising that an approach that provides predictions of the future, however imperfect, would have an advantage over reasoning about a situation based only on where objects are and how they were moving in the moment. The results of the investigation indicated that prediction provided a more effective appraisal of the value of an action, without requiring detailed rules.

There is a caveat here though: the evaluation of the heuristic model was an evaluation of its specific rule set, and it could have been developed further. Its rule set was not very complicated, and it is entirely possible that with a larger rule set and more detailed situational knowledge, it could have out-performed the predictive model. Indeed, both the heuristic and predictive models could have been developed further, to leapfrog each other in a virtual arms race. It is impossible to conclude that a predictive model would or could always outperform a purely heuristic model.

However, there was another aspect to the modelling. The predictive model only required 26 rules versus the 45 rules of the heuristic model. Not only were there less rules, they were simpler. Each rule simply stated a possible shot to test, and required no expert knowledge of how, or when, that shot might be used. In comparison, developing the heuristic rules required an understanding of squash strategy, and each rule had to be carefully considered as to how it would play out.

While both models could have been extended, the effort required to do so would have been considerably different. The heuristic model would require a lot of expert knowledge. The predictive model would have required only fixing some design issues and, perhaps, increasing the fidelity of the predictions. Of course, the predictive model does require a simulation engine that can predict outcomes of actions, however imperfectly. Developing the simulation does not require expert knowledge of squash either, but it does require being able to model the physics of the scenario. This is not an inconsiderable task and, even in the simple scenario used in this research, more time was spent developing the simulation than was required for the creation of the AI rule set.

VI. FUTURE WORK

The research described above only looked at a highly discrete problem, and the solution was very domain specific. The PACT-R cognitive model gave a scene description and predictions in a very squash-centric way. Continuing this methodology of creating a custom model and simulation for every scenario is time consuming, and it would be desirable to accelerate the process by finding a more generic way of describing physical relationships and actions within an environment.

It is unlikely that any solution could be truly generic. Such a solution would have to be able to model and simulate a large and arbitrary amount of the real world. Rather, a practical improved implementation of PACT-R would provide a generic framework that could be extended and adapted for specific scenarios.

PACT-R was intended for eventual use in robotics and embodied AI. However, taking this system into the real world presents the considerable challenge of perceiving and simulating at least a small part of the real world. For constrained situations this might not be so difficult. For example, in real-world squash, if you can detect and track the ball, it is then relatively easy to predict where it will go in the rectangular room that squash is played in. The bigger challenge would be predicting the outcome of shots, since this is not as clear-cut in the real world as it was in the simulation. The simulated shots were simplified, and the virtual robots were able to play them more consistently and accurately than any real human or robot would be able to. In a broader context the simulation could only be a probabilistic approximation of the real world and exploring what fidelity is needed to be functional is an area of possible future research.

The research also highlighted some issues when working with ACT-R that could be an interesting topic of future work. ACT-R's reinforcement learning mechanism did not work for this task since it would reinforce a single solution where a true squash player would always be choosing from a set of solutions. A flexible model would allow multiple solutions to a problem and some degree of random selection between them. In squash, as in many sports, there is an optimum decision based purely on the current circumstances, but it is not always the best choice to make if the opponent can predict the decision. Occasionally selecting a sub optimal solution can be an effective long term strategy. This is not something that is readily supported by ACT-R's reinforcement learning mechanisms.

In modelling within ACT-R pattern values, in rules, are tested with a basic set of comparative operators (>, <, =, etc.) While this is suitable for a lot of modelling, when implementing a squash strategy, it would have been convenient to have been able to model in fuzzy logic, where instead of yes/no answers, cold/cool/warm/hot answers were possible. The matching would bias the rule selection, rather than simply excluding or including specific rules. Giving ACT-R a fuzzy logic matching system would allow it to work better in situations where there is not a simple black or white answer.

ACT-R also has a declarative memory system (long term memory). This was tested after the main research was completed, but the results were poor due to the time penalty imposed by memory retrieval in ACT-R. There are additional issues related to the learning mechanism used in the memory system. That mechanism is based on a principle of spreading activation, where recently used memories are more likely to be recalled, and memories that share similar content are also more likely to be recalled (this is the spreading activation). This is not an appropriate mechanism for squash, since all shots and outcomes need to be considered equally. Despite these issues, the use of long term memory would seem to be a desirable feature, particularly as a key part of any cognitive system is learning. Sub symbolic learning is only one mechanism, episodic memory (stored in LTM) is another approach that should be considered in future versions.

If declarative memory had been used, how could it have been used, and what sort of learning mechanisms could have been applied? Could reinforcement learning be used with memories? Could there be negative and positive memories, a sort of 'positive memories' that are easily recalled, and 'negative memories' that are suppressed? These considerations may be crucial for applying simulation-based prediction in different robotic applications.

PACT-R respected ACT-R's timing for processing rules. This is based on human performance. Using this made testing and evaluation easier and more consistent. However, an artificial intelligence implemented on a computer does not need to be constrained by the limitations of human cognitive processing speeds. Likewise, the testing of actions in PACT-R was performed in a serial fashion. It would be possible to test multiple actions concurrently using common multiprocessing capabilities. These aspects remain to be tested and evaluated in a future version.

One possible negative aspect of PACT-R was that the cognitive decision process was partially outsourced to an external process. The cognitive rules choose a set of actions but the physics simulation made the final choice. This subverts the intent of a cognitive architecture like ACT-R where the cognitive model is expected to decide actions. Are there alternative approaches that could leverage the power of simulation to inform the model rather than bypass it?

The results obtained in this research suggest that using a physics engine within a cognitive architecture can simplify the cognitive modelling required in the decision making process although there were limits to the approach taken. Addressing those limitations in future research may lead to AIs based on cognitive architectures that are better suited to use in robotics.

REFERENCES

[1] D. Pentecost, C. Sennersten, R. Ollington, C. A. Lindley, and B. Kang, "Predictive ACT-R (PACT-R) Using A Physics Engine and Simulation for Physical Prediction in a Cognitive Architecture," in *COGNITIVE 2016 : The Eighth International Conference on Advanced Cognitive Technologies and Applications*, 2016, no. c, pp. 22–31.

[2] U. Kurup and C. Lebiere, "What can cognitive architectures do for robotics?," *Biol. Inspired Cogn. Archit.*, vol. 2, pp. 88–99, 2012.

[3] H. Q. Chong, A. H. Tan, and G. W. Ng, "Integrated cognitive architectures: A survey," *Artif. Intell. Rev.*, vol. 28, no. 2, pp. 103–130, 2007.

[4] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[5] W. DUCH, R. J. R. J. OENTARYO, and M. Pasquier, "Cognitive Architectures: Where do we go from here?," *Proc. 2008 Conf. Artif. Gen. Intell. 2008 Proc. First AGI Conf.*, vol. 171, no. OCtOBER, pp. 122–136, 2008.

[6] P. Jackson, *Introduction to expert systems*. Addison-Wesley Pub. Co.,Reading, MA, 1986.

[7] J. C. Giarratano and G. Riley, *Expert Systems: Principles and Programming*. PWS Publishing Co., 1998.

[8] A. Ajith, "Rule-based Expert Systems HEURISTICS," *Handb. Meas. Syst. Des.*, vol. 1, no. g, pp. 909–919, 2005.

[9] C. A. Lindley, "Synthetic Intelligence: Beyond Artificial Intelligence and Robotics," in *Integral Biomathics*, Springer,

2012, pp. 195–204.

[10] C. A. Lindley, "Neurobiological Computation and Synthetic Intelligence," in *Computing Nature*, 2013, pp. 71–85.

[11] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 110, no. 45, pp. 18327–32, 2013.

[12] J. R. Anderson, J. M. Fincham, Y. Qin, and A. Stocco, "A central circuit of the mind," *Trends Cogn. Sci.*, vol. 12, no. March, pp. 136–143, 2008.

[13] J. R. Anderson and C. D. Schunn, "Implications of the ACT-R learning theory: No magic bullets," *Adv. Instr. Psychol. (Vol. 5)*, vol. 5, pp. 1–34, 2000.

[14] J. R. Anderson, *How Can the Human MindOccur in the Physical Universe?* Oxford University Press, 2007.

[15] H. Putnam, "Brains and Behavior," *American Association for the Advancement of Science*, vol. Section L. 1961.

[16] S. Profanter, "Cognitive architectures," *HaupteSeminar Hum. Robot Interact.*, p. 27, 2012.

[17] S. Löwel and W. Singer, "Selection of Intrinsic Horizontal Connections in the Visual Cortex by Correlated Neuronal Activity," *Science (80-.).*, vol. 255, no. 5041, pp. 209–212, 1992.

[18] J. Laird, K. Kinkade, S. Mohan, and J. Xu, "Cognitive Robotics Using the Soar Cognitive Architecture," *8th Int. Work. Cogn. Robot.*, pp. 46–54, 2012.

[19] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *IEEE Trans. Evol. Comput.*, vol. 11, no. 2, pp. 151–180, 2007.

[20] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.*, vol. 2, no. 1, pp. 14–23, 1986.

[21] R. a. Brooks, "Elephants don't play chess," *Rob. Auton. Syst.*, vol. 6, no. 1–2, pp. 3–15, 1990.

[22] R. A. Brooks, C. Breazeal, M. Marjanovi, B. Scassellati, and M. M. Williamson, "The Cog Project : Building a Humanoid Robot," in *Computation for metaphors, analogy, and agents*, Springer Berlin Heidelberg, 1999, pp. 52–87.

[23] M. Minsky, R. Kurzweil, and S. Mann, "Society of mind," *Artificial Intelligence*, vol. 48, no. 3. pp. 371–396, 1991.

[24] P. Singh, "Examing the Society Of Mind," *Comput. Informatics*, vol. 22, pp. 1001–1023, 2003.

[25] T. Liadal, "ACT-R A cognitive architecture," *Univ. des Saarlandes*, pp. 1–16, 2007.

[26] M. Lochner, C. Sennersten, A. Morshed, and C. Lindley, "Modelling Spatial Understanding : Using Knowledge Representation to Enable Spatial Awareness in a Robotics Platform," vol. 7, no. c, pp. 26–31, 2014.

[27] C. Sennersten, A. Morshed, M. Lochner, and C. Lindley, "Towards a Cloud-Based Architecture for 3D Object Comprehension in Cognitive Robotics," no. CCI, pp. 220–225, 2014.