THE DESIGN OF PROCEDURAL, SEMANTIC, AND EPISODIC MEMORY SYSTEMS FOR A COGNITIVE ROBOT

By

Will Dodd

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

August, 2005

Nashville, Tennessee

Approved:

Professor Kazuhiko Kawamura

Professor David C. Noelle

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Kazuhiko Kawamura, for his sage guidance and support. I would not have been able to compile this thesis without him, and he has provided me with insight and direction since the day I first started graduate school.

I would also like to thank Dr. David Noelle and Dr. Mitch Wilkes for helping me learn that knowing how to ask the correct question in a tough situation is sometimes more important than finding its answer.

Flo Fottrell deserves my thanks for helping me throughout my graduate career. She has assisted me in finding my way around the school, and always gives the right advice.

Similarly, I would like to acknowledge my fellow students in the CIS for all their help, especially when it came to developing my Linux administrator skills.

My parents receive my deepest gratitude for their rock-solid support and for instilling in me a love of knowledge, and my brother for teaching me humility of that knowledge.

Finally, I would like to thank Shazi Jiang for standing beside me throughout my MS career. I couldn't have done it without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

One of the main challenges of robotics is the production of intelligent behavior in complex domains [Fulton and Pransky, 2004]. Many approaches to this end have been developed, from subsumption architectures to reinforcement learning techniques. The success of these approaches, however, has been limited by the inherent complexity of interaction with the real world – such approaches work relatively well until a new situation or unrecognized stimulus is encountered. At that point, irrational behavior is often encountered.

Another facet of artificial intelligence that has met with limited success is that of the emulation of human cognitive performance by a computer system for real-world tasks in complex domains.

For example, the Digital Aristotle project [Friedland et al., 2004] has attempted to develop a system that is able to pass a chemistry test given over 70 pages of material at the high school level. Many resources have been spent on this highly domain-specific system, and only three out of five of the expert systems developed were determined to have the capacity to match human performance on high school AP Chemistry tests. These evaluations required that the engineers who developed the system change the format of the questions from English to a system-specific format. In other words, in a relatively restricted environment, with specially designed systems and at a cost of over $10,000 a page, the developed systems met with limited success.

Given the weaknesses of these approaches when dealing with complex, dynamic environments, perhaps a valid alternative lies in the development of embodied cognitive systems. Such systems, inspired by human psychological and neuroscientific performance, have the capability of learning effective and dynamic behaviors in complex domains in which more rigidly designed techniques fail.

These systems also have the capability to enhance the artificial intelligence techniques used to build them through the concept of embodiment [Brooks, 1991]. Because the system is able to learn from and interact with its environment, it is able to actively learn instead of merely attempting to fit percepts with preprogrammed rules as is seen in the Digital Aristotle system. This embodiment is an important characteristic in the development of social, intelligent systems [Mataric, 1997].

One crucial consideration when creating a cognitive system is the design of the memory system. The memory of the cognitive system empowers the system to learn from its environment and, more importantly, from past experience. As such, the memory system must be carefully designed in order to maximize the potential of the cognitive system. It is important that the memory system not only be able to recall information that will likely be needed in the everyday operation of the system, but also that the system do so in a timely manner. Because the system is embodied, it must interact with a dynamic environment sometimes requiring that responses are timely in order to be relevant.

The following document details the design of such a memory system for a robotic system. The memory is based on the memory system of a human, with both short-term, which holds percepts of the robot, and long-term, which holds information obtained from past experience, memory systems. The long term memory is further divided into three sections: Procedural Memory, which holds information needed to execute movements by

the robot, Semantic Memory, which contains factual information not associated with a particular time and place, and Episodic Memory, which stores specific experiences of the system. A Working Memory System is also designed to hold task-relevant information from the long-term memories.

Similar divisions in human memory are generally accepted in the field of Psychology [Tulving, 1985], with some contention on the difference in classification between Short-Term and Working Memory. For the purposes of this Thesis, it is assumed that Working Memory is an active storage space containing task-relevant information while the Short-Term Memory holds information representing the current percepts of the system.

In this thesis, the term Working Memory System is meant to denote the complete system that retrieves memories from the memory stores based on the current context. The individual working memory systems are denoted by Memory Name – Working Memory, for example Procedural Memory – Working Memory.

Objectives

The objective of this work is to create a memory system modeled on the above division from human psychology for application to a humanoid robot named ISAC which is located in the Center for Intelligent Systems at Vanderbilt University.

This memory system should be flexible enough to represent information needed for solving problems in ISAC's daily operation, learning solutions to tasks from humans, generating movement commands to move ISAC's actuators, and perceiving objects that might be used to achieve tasks in ISAC's operation.

The memory systems should balance this flexibility to be specific enough so that retrieval can be accomplished in a sufficiently accurate and timely manner. The Working Memory System of ISAC should function in a manner that is relatively automatic from the perspective of the systems that manage goals, plans, and task execution. This requires that each memory system have some method of evaluating the current context for retrieval of correct memories, and that these memories be designed in a manner that facilitates this evaluation.

Significance of Work

This work is significant because it details the creation of a system of the three major human memory classifications and application to a robotic system. The creation of the Episodic Memory System is especially significant because the creation of a general purpose Episodic Memory and Episodic Memory retrieval system has been relatively ignored in the symbolic AI community [Nuxoll and Laird, 2004]. As such, the application of this type of memory to a cognitive robotic AI system is novel.

Also, the development of a separate adaptive working memory system that acts semi-autonomously to retrieve appropriate memories based on the current context is a relatively novel concept within a symbolic cognitive system.

Summary and Organization

Chapter II details background information on several cognitive systems with their respective memory systems, adaptive working memory, and ISAC's cognitive system. This chapter is intended to give a survey of appropriate literature as well as an overview of the system into which the topics of the following chapters will be integrated. Chapter

III discusses the Procedural Memory system for generating movement on ISAC. Chapter IV talks about ISAC's Semantic Memory, while Chapter V discusses the Episodic Memory. Chapter VI details the experimental setup for experiments to test the memory systems described in Chapters III-V, and Chapter VII discusses the results of these experiments and their implications for the design and use of memory systems. Chapter VIII gives some directions in which this research may be taken.

CHAPTER II

BACKGROUND MATERIAL AND PREVIOUS WORK

Robotics researchers have attempted to apply many different Artificial Intelligence (AI) techniques to robotic control and learning. Numerous attempts at achieving human-like intelligence through the use of both traditional and probabilistic AI have revealed the limitations inherent in each approach. One recent approach to the problem of generating dynamic, robust robotic behavior has been that of combining traditional AI with computational neuroscience and psychological research to produce a system that more closely attempts to emulate human performance on tasks.

The following section details several systems designed by cognitive theorists attempting to recreate human mental performance either in the context of interactions with the real world or in a laboratory in order to test psychological theories. These systems attempt to recreate human cognition and are often called "cognitive systems" or "cognitive architectures". The term "cognitive agent", on the other hand, is intended to denote a cognitive architecture that is embodied and exhibits goals and desires in its interactions with its environment, while the phrase "cognitive robot" is used to describe a cognitive agent that is intended to be implemented within a robotic system. Two systems that are designed to be directly implemented on a cognitive robots are detailed in the "Cognitive Agent" section, and two more general purpose architectures are detailed in the "Cognitive Architectures" section.

The two systems described in the Cognitive Architectures section, SOAR and ACT-R, were selected over other systems because they are the most widespread

cognitive systems intended for research in existence. The two systems described in the Cognitive Agent section, IDA and Haikonen's architecture, are chosen to reflect two very different methods of attempting to apply a model of human cognition to the control of an embodied agent. Haikonen's method uses statistical AI while Franklin's IDA employs a more symbolic approach.

Cognitive Systems and Memory

Cognition in humans is exhibited through the characteristics of "short and long term memory, categorizing and conceptualizing, reasoning, planning, problem solving, learning, [and] creativity" [Franklin and Graesser, 1997]. A cognitive system employs a combination of some or all such characteristics listed above within a structure derived from cognitive psychology and computational neuroscience.

"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future" [Franklin and Graesser, 1997]. A cognitive agent or cognitive machine is an embodied autonomous agent that exhibits the characteristics of a cognitive system. Figure 1 shows the structure of a generic cognitive agent as defined by Franklin.

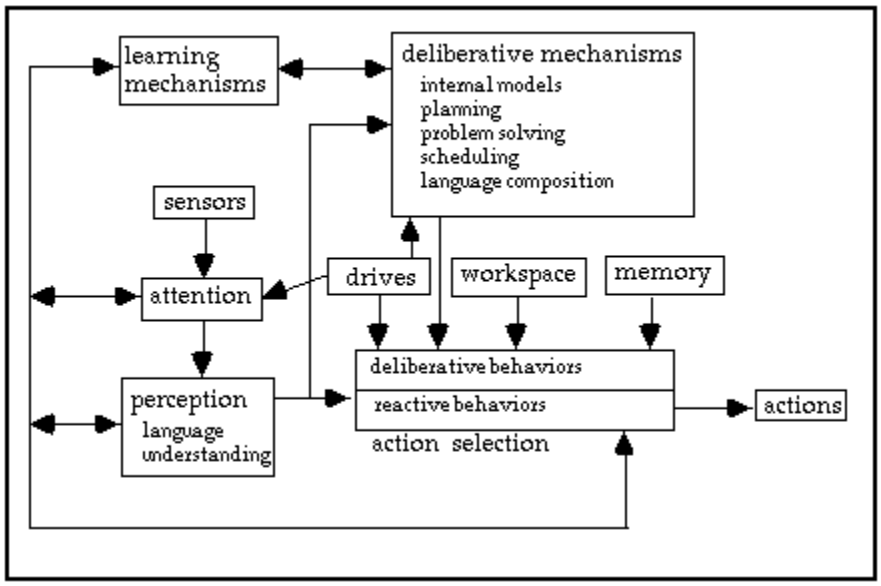

Figure 1. Conceptual Diagram of a Cognitive Agent with Learning [Franklin, 1997]

Figure 1 shows a representation of Franklin's view of a generic cognitive agent based on the action selection paradigm of mind, in which the function of the cognitive system is to produce an action based on perceptions from the environment. The action selection paradigm of mind dictates that the mind is an aggregate of many simple processes that have the end goal of selecting the next action for an organism [Franklin, 1997]. When implemented into a cognitive agent, this theory may be conceptualized as a system like that shown in Figure 1.

An important characteristic of such a cognitive agent is *embodiment*. The agent must be able to both perceive and act, or be "structurally coupled", in an environment in order to produce cognitive behavior. This is because, in the action selection paradigm of mind, the function of a cognitive system is to generate actions based on perceptions from the environment. Without embodiment, the agent can neither perceive the environment or act on that environment..

Haikonen, on the other hand, suggests a somewhat different definition of a cognitive machine, a term meaning a physically embodied cognitive agent. According to him, a cognitive machine should exhibit several cognitive processes shown by humans in addition to characteristics such as those listed by Franklin. These include the capacity to sense its environment, contain a properly-grounded flow of inner speech and imagery, and display the effects of attention [Haikonen, 2000a]. Although these concepts are not identical to Franklin's definition, they are complementary. This may be seen by analyzing Figure 1, which incorporates all three characteristics -- inner speech and imagery would be contained within the deliberative processes and workspace.

The developers of Soar, a common cognitive architecture used for modeling human psychological performance, define a cognitive architecture to be “a theory about the fixed computational structure of cognition” [International Encyclopedia of the Social and Behavioral Sciences, 2005]. Soar consists of a flexible architecture for computational psychology as opposed to a monolithic system built for a specific purpose.

The creators of ACT-R, another flexible cognitive architecture designed for use with computational psychology, state that a cognitive architecture is a “theory about how human cognition works” [Budiu, 2005]. ACT-R was created on the theory that an integrated cognitive system is far more valuable for developing an understanding of the human mind than a system intended to perform a specific psychological task [Anderson et al., 2004]. ACT-R appears to the user as a programming language, but is actually a model based on human psychological experiments and assumptions about human cognition.

Finally, the Foresight Project, a project in the UK intended in part to research cognitive systems, puts forth perhaps the broadest definition of a cognitive system: “Cognitive systems are natural or artificial information processing systems, including

those responsible for perception, learning, reasoning, decision-making, communication and action" [Foresight, 2005]. This definition equates cognitive systems with artificial intelligence, however the definition proposed by [Franklin, 1997] will be used in this thesis.

General Purpose Cognitive Systems

The cognitive systems discussed in the following section are primarily designed to explain human psychological phenomena. While they are not specifically designed for inclusion on embodied agents such as robots, much can be learned from these architectures that emulate humans. One of the main uses for these architectures has been to model and conduct psychological experiments. Everything from human memory learning techniques [Pavlik and Anderson, 2004] to language performance [Budiu and Anderson, 2003] has been evaluated with these systems. This shows the flexibility of the system, however this flexibility comes at the cost of specialization of components, including memory structure.

One striking characteristic of Soar and ACT-R when compared with the more specialized cognitive agents is the lack of recognizable memory structure. Both systems use production rules and predicates to produce desired behavior, and neither employs a rigid memory hierarchy. Memory design is left to the designer of the system, as long as certain system-level restrictions are followed.

Soar Cognitive Architecture

The first architecture to be discussed is Soar. Soar is a general purpose architecture that has been implemented to solve problems from psychological analysis to

robotic control. It is important to note, however, that the Soar system is a framework and not a specific system designed for a particular task.

The Soar cognitive architecture was designed to be a flexible framework that could serve many purposes for the simulation of human cognitive activities. All operations in Soar consist of applying operators to states in pursuit of some goal using problem space search. Regarding the structure of Soar: “Together, the task-implementation and search-control functions are sufficient for problem space search to occur. The quality and efficiency of the problem solving will depend on the nature of the selected functions." [Laird et al., 1987]. In other words, the system is competent for solving problems, however it is only as good as the information that is made available to it and rules provided to it.

Figure 2 shows the structure of the Soar Cognitive Architecture. Soar's Long Term Memory is entirely composed of production rules. These rules require a pattern of preconditions from the Working Memory system and a set of actions to perform when the production's preconditions are fulfilled. For example, a simple rule might be: “If a person is identified, then greet them.” The precondition to this rule would be the identification of a person in the agent's environment. The action to be performed would be the greeting action. Perceptions are posted to a Working Memory storage space, managed by the preference system. This storage system is of indefinite size, and grows with the problem since production rules can post results to working memory in the hopes of satisfying more preconditions for actions.

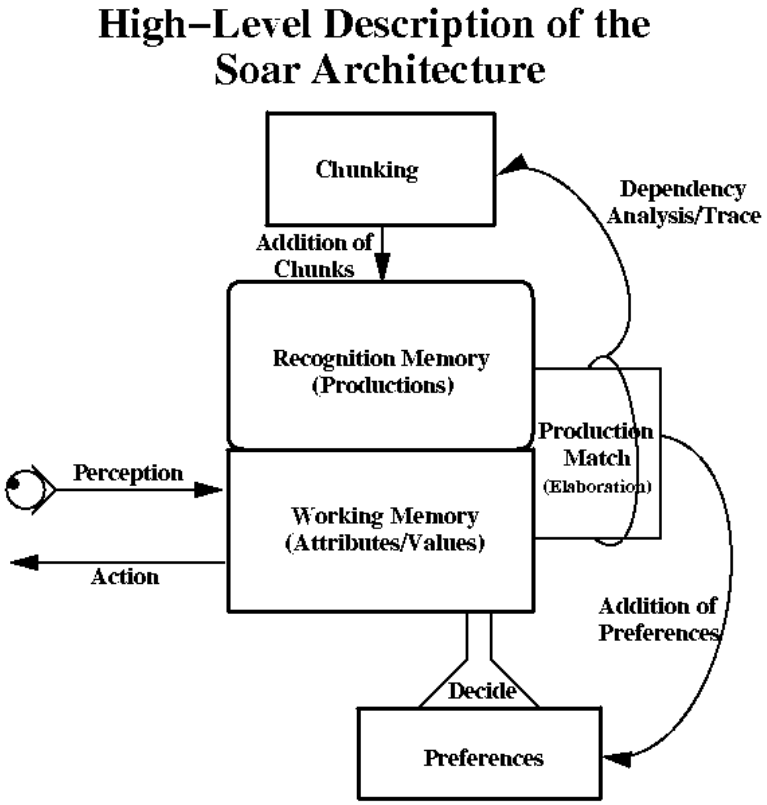

Figure 2. Soar Cognitive Architecture [modified from Wray et al., 2005]

Soar has two phases of operation: elaboration and decision [Wray et al., 2005]. During the elaboration phase, the contents of the Working Memory are matched with production rules so every possible match is asserted. The rules are then fired in parallel, and the only action a fired rule can take is to add a value-attribute pair to the Preference Memory. The rules keep firing until no more firing is possible.

Superficially, it may be seen that Soar contains two types of memories, rules and attributes or values that can satisfy and affect rules. Without discussing systems built on top of Soar, several parallels may be drawn from Soar to the human memory division discussed in Chapter I.

The Working Memory of Soar is different than the working memory described in detail in this thesis. In Soar, the working memory holds all information in the current problem space, while the working memory discussed in this thesis holds only current task and context-relevant information. The working memory in both cases does, however, hold information that is being considered for use when producing actions or changes in internal state.

Semantic memory can be attributed to all attributes and values. These represent independent facts, and are used to produce action within the system. These memory units are not necessarily derived from perceptions, but can also be produced by the firing of rules.

Procedural memory, in turn, can be linked to the production rules in Soar's Recognition Memory. These rules exist to change either the internal state of Soar or the state of the environment. Regardless, these rules accomplish actions and may be likened to the procedural memory discussed in Chapter I.

Although episodic memory systems have been built under Soar's architecture [Nuxoll and Laird, 2004], episodic information is not readily obtained from Soar's base system. In particular, no memory units are associated with a specific time and place, or are represented in a format exhibiting characteristics attributed to episodic memory.

ACT-R Cognitive Architecture

ACT-R is similar to Soar in terms of memory design. Production rules are contained within a procedural memory system, while semantic memories are used for matching, and then firing, productions.

ACT-R stands for the Adaptive Character of Thought – Rational. ACT-R is a cognitive system used extensively in psychology for analysis of human performance on various psychological tasks, although it draws much of its inspiration from observations about rational behavior (rationality in a human-like genetically-based system is defined as the action that maximizes an organism's chance of reproducing).

ACT-R consists of three basic components: modules, buffers, and a pattern matcher [Budiu, 2005]. These components are combined to produce a system intended to be analogous to certain aspects of the human brain – namely those aspects in which the experimenter is interested.

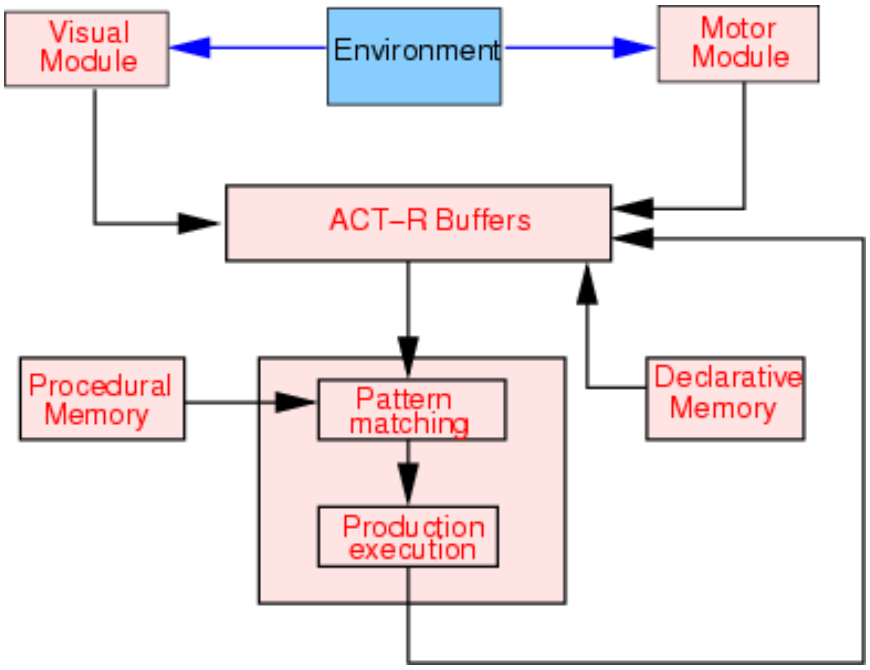

Figure 3: Functional Structure of ACT-R [Budiu, 2005]

There are two types of modules: perceptual-motor and memory. As the name suggests, perceptual-motor modules interact with the real world by generating percepts or actions.

The memory modules are subdivided into two categories: declarative and procedural. The declarative memory holds declarative facts. The procedural memory holds productions that are used to pursue actions. These productions hold the same IF-THEN form of the SOAR production rules. An example of a production rule would be "IF the red beanbag is on the table, THEN pick it up." These modules are used by the system shown in Figure 3 and are the atomic components that are passed around the system when working towards a specific goal.

Buffers hold declarative memory modules to be used by the system, and are somewhat analogous to human working memory systems [Anderson et al., 2004]. The state of the system can be shown by the content of the buffers, allowing for easy system debugging and design.

The final major component of the ACT-R system is the pattern matcher. This analyzes the contents of the buffers and attempts to match a production rule to them. Unlike Soar, only one production rule can be fired in each time step, allowing the system to work in an easily traceable manner. Although the execution of a Soar task may be retraced by recording all fired rules, if the system executes one action ever time step it is easier to apply a case-based reasoning model to a record of task execution stored in a structure analogous to episodic memory.

The system described so far is the symbolic component of the ACT-R system. There is another layer that is used for learning and conflict resolution called the subsymbolic system. The subsymbolic system runs in parallel with the symbolic system. Many facets of the subsymbolic system run in parallel to influence the performance of the symbolic system.

The subsymbolic system implements a complex set of formulas that do things like judge the cost to the system of firing a certain production to evaluate multiple satisfied productions, or associating costs and time delays with the retrieval of memory contents. For example, if lifting the blue box has a 10% chance of disabling an actuator, but lifting the red box has no chance of destroying the actuator, firing the rule that lifts the blue box would be undesirable. This level of desire or cost could be implemented in the subsymbolic system to influence the behavior of the system without having to create explicit production rules or preconditions. The subsymbolic system provides the designer of the system with an extra level of control in the reproduction of human behavior.

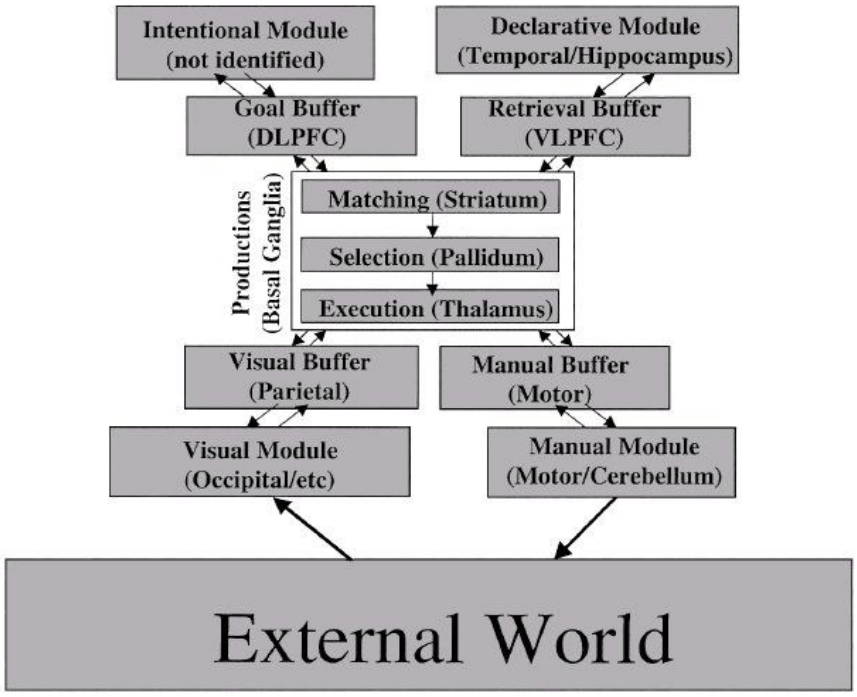

Figure 4. ACT-R and Human Brain Correlates [Anderson et al., 2004]

Figure 4 shows the structure of the ACT-R architecture when compared with the human brain. The areas of the brain thought to be responsible for each element of ACT-R

are given in parenthesis beneath the module name. By drawing such correlates, ACT-R gives researchers a powerful tool to use when evaluating human performance on tasks. It also allows for the subsymbolic layer to be adjusted to match neurological data available for specific functions and physical structures.

Many of the same comparisons that were drawn between Soar and the memory types discussed in Chapter I may be drawn between ACT-R and these same memory classes.

For example, the Procedural Memory in ACT-R is intended to be analogous to the procedural memory class of a human. Similarly, the Declarative Memory unit of ACT-R is similar in function to the semantic memory store in a human.

The buffer contents of the system represent a sort of working memory system, although they do not hold any reference to the sub-symbolic system, which influences task execution a great deal. It is important to note that the contents of the buffer system are those that influence the firing of Procedural Memory rules, just as the contents of the working memory of a cognitive system influence the conscious reasoning process of that system.

Cognitive Agents

The following cognitive system architectures are designed to be embodied cognitive systems, or cognitive machines, and are meant to interact intelligently with their environments in a robust, adaptive manner. These two systems were selected for review because they are embodied systems intended for implementation in specific environments, similar to ISAC's cognitive system. Franklin's IDA was selected for its various memory structures as well as its use of symbolic memory structures.

These architectures could be successfully integrated onto a robotic platform due to their robust design, and are labeled embodied cognitive systems to separate them from more general-purpose cognitive architectures such as Soar or ACT-R which are intended to be adapted to a wide range of applications, from testing psychological theories to controlling complex systems.

IDA: A Cognitive Agent Architecture

A system of interest to the design of symbolic memory structures for an embodied cognitive system is IDA, an embodied cognitive agent using several different types of symbolic memory structures. IDA was designed to reassign sailors in the United States Navy to jobs based on their own requirements and the needs of the navy. This architecture is intended to demonstrate the utility of a software cognitive agent, and was intended to replace the Navy's $20 million/year human-based system [Franklin et al., 1998].

Figure 5 shows the system architecture of IDA. It is a specific implementation of the generalized cognitive agent framework detailed in Figure 1. IDA is intended to implement a theory of consciousness called “Global Workspace Theory” and developed by Baars [Baars, 1988]. Perception is carried out by semantic networks called slipnets, the Behavior Net selects actions, and a knowledge base handles the deliberative aspect of a cognitive machine.

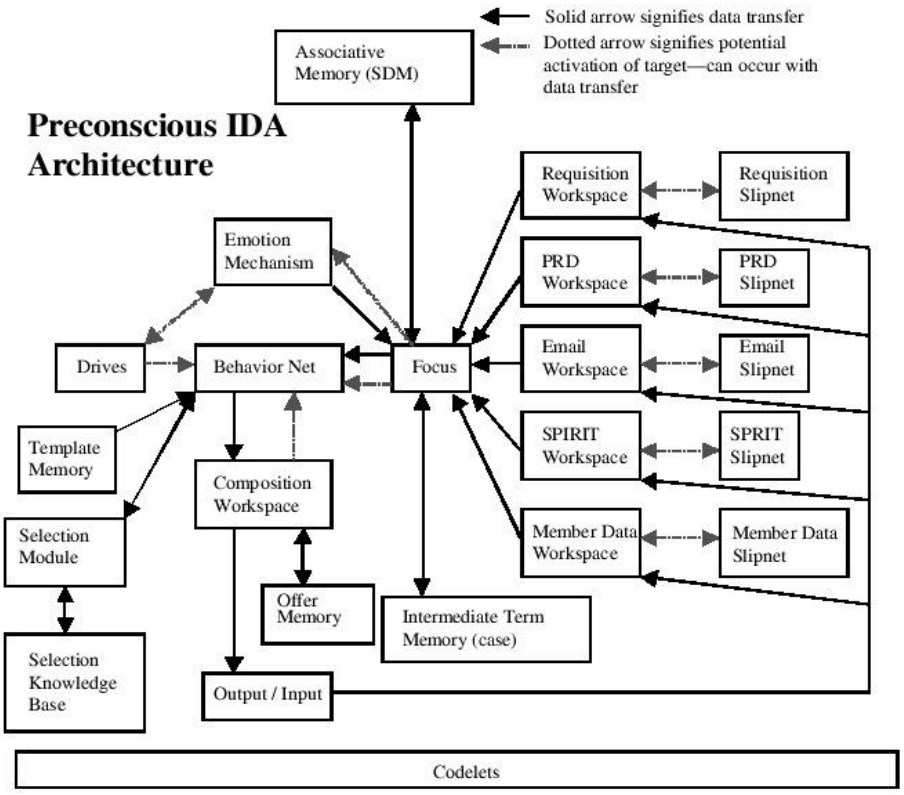

Figure 5. Architecture of IDA, a Cognitive Agent [Franklin et al., 1998]

The slipnets and Behavior Net consist of systems of codelets, or processes, that each perform a simple task. The codelets are activated by a spreading activation network, and post their results in a global workspace [Franklin, 1995]. While the spreading activation network is hard-coded, the workspace holding the current percepts or plan of action is fluid. Codelets can also post activation to the spreading activation network. This allows the system to adapt to changing situations and maintain an internal activation flow.

Even though the application for IDA is limited to billet assignment, it has already proved to be a valuable tool for analysis of human cognition and consciousness [Baars and Franklin, 2003] [Franklin, 1999]. A current implementation of IDA is running successfully for assigning billets to sailors [Franklin and Graesser, 2001], although it does not contain those elements needed for implementation of Baars' theory of consciousness.

IDA has four types of memory: associative, offer, template, and intermediate. The associative memory attempts to associate sensory inputs with actions, and is modeled after Kanerva's sparse distributed memory [Kanerva, 1988]. Offer memory keeps track of offers given to sailors, and is implemented with a traditional database. The template memory stores various templates needed to interface with email systems, databases, etc. Intermediate memory serves as an intermediate-term case-based episodic memory system, and serves to store the emotional output of the system, the content of the perception registers, and the action taken [Franklin et al., 1998].

The Focus is a memory cache that stores a single input from each perception and memory system to which it is connected. This memory store serves as a bottleneck to simplify the interface to the Behavior Net and memory stores, and to make the system function in a step-wise fashion. In this manner, it simplifies the interfacing of the system by reducing complexity in time and information.

Although this system does not directly parallel the psychological division of memory detailed in Chapter I, several parallels can be drawn between its memory structures and those that were outlined.

Perhaps the most obvious similarity is that IDA's Intermediate memory can be compared with the Episodic Memory of a human. The Intermediate memory stores perceptions, actions, and a representation of the internal state – the emotional output of the system. Although it is used to provide context to a particular transaction, in a more complex domain perhaps such a memory could be extended to allow for case-based reasoning, or comparison of separate experiences through recorded episodes.

The Offer and Associative memories are Semantic in nature. Although they represent facts, those facts are disassociated from a particular time or place, or at least a

specific sequence of experiences undertaken at a particular time or place in the case that the database entry had a timestamp. The Associative Memory is implemented with a reconstructive retrieval process that attempts to recreate memories based on a cue instead of directly retrieving them from a database using Kanerva's Sparse Distributed Memory technique [Kanerva, 1998]. This process maps the memory to a large binary vector. Memories are then retrieved by a technique that recursively retrieves and reconstructs values until a steady result is reached. If the cue is within the attractor basin for a given memory, that memory will eventually be retrieved.

The Template memory could be considered to be comparable to Procedural Memory. The system interacts with its environment, turning some sort of internal representation into the action taken by the system. In this case, the internal action is hard-coded in the behavior network, while the translation of this command to an external action is characterized by the contents of this memory network.

Franklin's Focus offers some of the same advantages as a working memory system (See the Working Memory section in Chapter II). It streamlines the operation of the executive components of the system by preventing those elements from having to sort through massive amounts of information. It also allows for an easy method of simplifying the debugging and analysis of a system built for such a specific, critical purpose.

Another way to view the Focus is as a sort of Short-Term memory, albeit a limited one. The Focus receives input from the perceptual elements of the system (shown to the right of the Focus in Figure 5) and represents the most important perception from the environment of this agent. Percepts must travel through the focus before they can influence behavior.

One important observation that one can draw from this viewpoint is the similarity between the function of hard-coded aspects of the system and the memory units that they employ. For example, the behavior network is hard-coded, but the links between its units could be viewed as a static example of a Procedural Memory. The decision of encoding parts of the system in dynamic memory units and others in static programs is a choice made by the system designer when balancing the need for that particular function of the system to evolve with experience and the requirement that the system be stable and efficient.

Haikonen's Neural Cognitive System Model

The second cognitive agent to be analyzed is Haikonen's general cognitive agent architecture. Although this architecture does not use symbolic AI memory units, it does contain some characteristics that show how a more statistical approach to cognitive system design can produce memory divisions similar to those listed in Chapter I. Haikonen's architecture is described in the cognitive agent section due to its inherently embodied nature.

Haikonen takes a somewhat different approach to building an embodied cognitive architecture. Instead of building specialized systems based on explicitly-programmed blocks, Haikonen developed a recurrent neural-network building-block that can be generalized to produce a cognitive machine [Haikonen, 2000b].

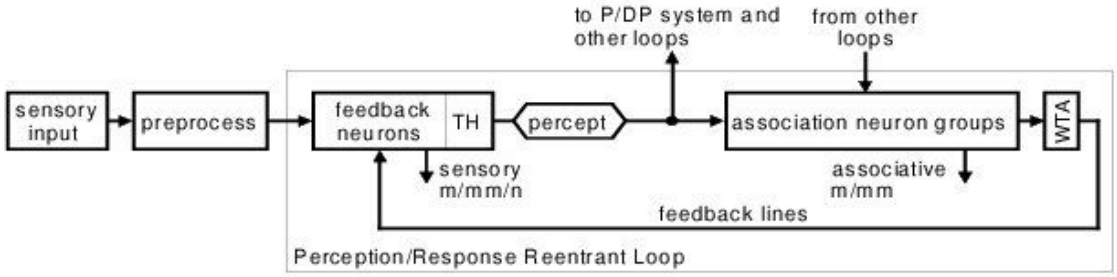

Figure 6. Haikonen's Perception/Response Reentrant Loop [Haikonen, 2000a]

Figure 6 shows the main building block of Haikonen's cognitive architecture [Haikonen, 2000a]. The sensory input contains sensory information from a specific modality. It is fed through the feedback neuron group, which combines attentional input from the association groups with perceptual information from the preprocessing network. The output from the feedback neuron group contains the *percept* of the perceptual block, which is the main output of the loop and is transmitted to the association blocks of the rest of the system. The sensory m/mm/n signal shows if the percept matches the output of the association group (m), mis-matches the association output (mm), or is a novel stimulus (n). In a similar way, the associative m/mm output contains information on whether the percept matches the output of other reentrant loops. Finally, a winner-take-all system selects the dominant output from the association neurons and sends it to the feedback neuron group.

Motor commands may be executed by sending the *percept* of the group to a motor command unit. In this case, the sensory input is proprioceptive information, so a feedback motor loop is created [Haikonen, 2003].

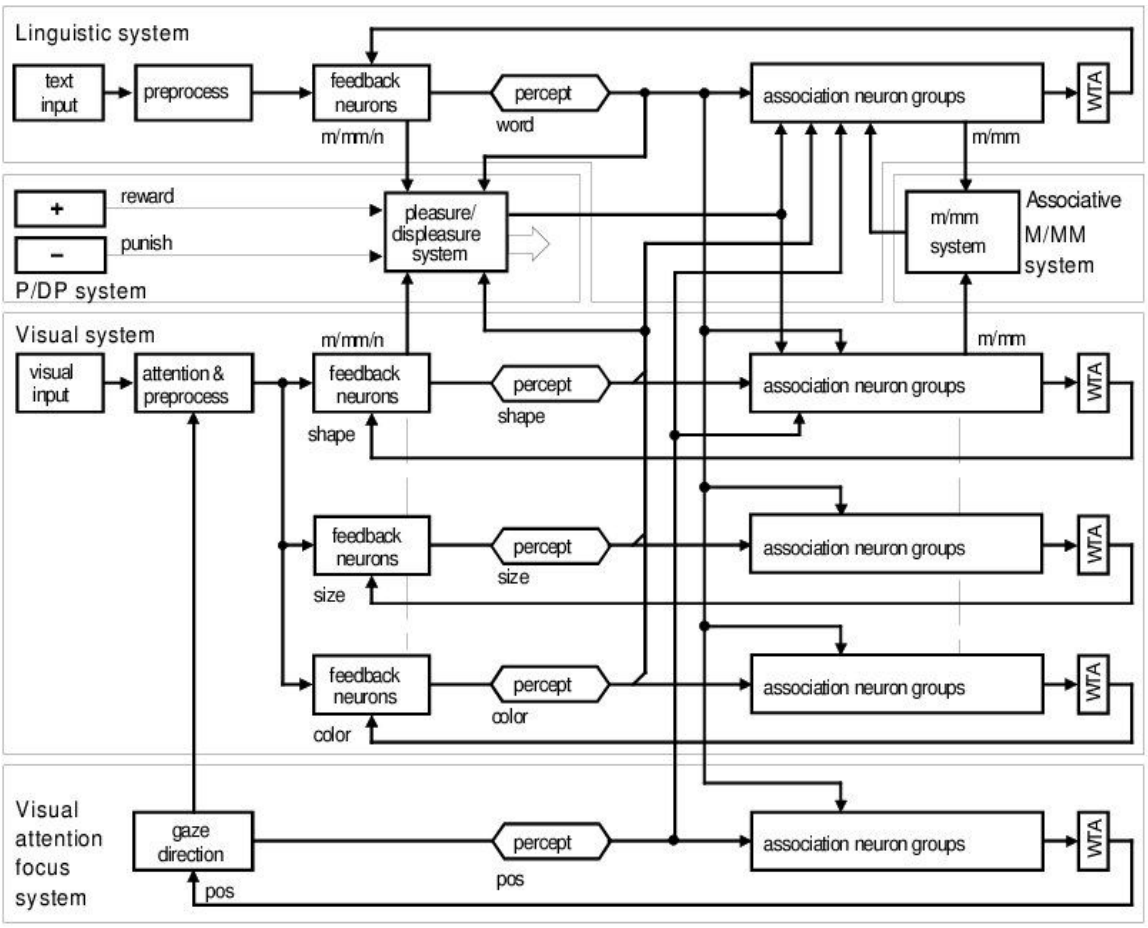

Figure 7. Cognitive System by Haikonen [Haikonen, 2000b]

Figure 7 shows a cognitive machine using five reentrant perception loops (one auditory, three visual, and one controlling gaze) and one reentrant system evaluation loop. A system such as this could be used for identifying stimuli and context. The m/mm/n system is intended to coordinate responses based on stimuli, system state, and percepts. The p/dp block associates emotional salience with percepts and changes the motor output of the system [Haikonen, 2000b]. Haikonen's emotion system is discussed more in Chapter V.

Due to the highly modular architecture of Haikonen's cognitive systems, the architecture is easily adaptable to a wide variety of applications. The architecture also

dictates that sensors are closely tied with the reward estimation and action execution system, so perception becomes intrinsically tied with cognition.

Like IDA, several characteristics of this system contain parallels to the memory division discussed in Chapter I. Each of these memory types is represented by the weights of the connections within the neural networks in each block of Figure 7.

Short-term memory could be contained within the output of the preprocessing block. These memories represent the current percept of the system and are transient in nature – they disappear once the stimulus leaves the perceptual field of the system.

The reentrant loop within each perceptual block represents a sort of working memory for the system. This loop contains an internal representation of relevant information within the system, and can be influenced by the other perceptual blocks as well as the reward systems. It also could be thought of as a short-term memory in certain circumstances, such as when the output of another perceptual block produces the pattern of activation representing the other block's perception.

Procedural memory is represented by the connection weights within the visual motor block that sends commands to the outside world.

The Semantic Memory of the system could be attributed to the association strengths between a particular block and the perceptual output of other blocks, the pleasure/displeasure and novelty systems, or even the preprocessing block and the perceptual loop itself. The semantic memory in this case serves to associate certain patterns with others, producing a “fact” of relation.

There is no episodic memory that can be attributed to this system, other than recurrent activation patterns. The reason the associations between a perceptual block and the pleasure/displeasure system can not be compared to an episodic memory system is that the connections are not associated with a particular time and place.

Discussion of Memory in Cognitive Systems

Several similarities have been drawn between the memory structures within these cognitive systems and those present in humans. One glaring hole, however, is the lack of an episodic memory system for general system-level recording and learning. IDA contains some episodic memory capability, however its treatment of episodic memory disregards many of the purposes for which humans use episodic memory.

One important difference between IDA's Intermediate memory and the episodic memory of a human is that the Intermediate memory is not stored past a particular interaction with a sailor. One of the main purposes of human episodic memory is to recall episodes that happened outside the current timeframe to remember relevant information about the present. For example, episodic memory might be used to string together landmarks used to navigate when visiting a novel city or to recall the place where one lost a credit card in a night around town. Neither of these tasks would be possible were episodic memory discarded after the current task was finished. As such, Franklin disregards retrieval or evaluation of Intermediate memory for context.

Another useful characteristic of the memory parallels drawn in the previous sections is that the short-term and semantic memories share a similar representation. Even in Haikonen's model, the short-term memory was transferred into a semantic memory as soon as it was perceived. For the semantic memory system detailed in Chapter IV,

semantic memory units are created by perceptual algorithms, and passed around the short-term and episodic memories as a common internal representation. This keeps the representations of memories consistent, and allows the system to derive statistics useful for memory retrieval and formation from these common units of knowledge.

Working Memory In Cognitive Systems

Working memory is a limited-capacity memory store that actively selects and holds the most task-relevant information in a cognitive system [Skubic et al., 2004]. Psychological and neurological studies provide much evidence for the existence of a working memory in humans [Baddeley, 1986][Goldmann-Rakic, 1987][Cowan, 2001]. Such a system could provide many advantages to cognition in both humans and artificial cognitive systems.

For the purpose of creating artificial cognitive systems, the most useful characteristic of working memory is that it automatically provides task-related focus. Since the environment is filtered and only task-related information is allowed to pass through to the cognitive processes, conscious resources are not wasted actively searching for relevant information but are reserved for task execution and monitoring. Working memory also enhances cognitive ability by reducing the complexity of the task-related space to several "chunks" that are needed for immediate decision making in the current context. The most relevant task-related information is selected, producing a vastly-simplified space in which the cognitive system pursues goals.

A "chunk" is the memory unit that is used by the working memory. This could be a single number, the relationship of a chess piece to others on the board, the location of an object, or a formative plan. The design of the working memory system and that of the

memory systems that provide the working memory system with data dictates the content of a “chunk” in an artificial cognitive system.

In humans, neurons of the prefrontal cortex are thought to be involved in the selection and maintenance of working memory contents [Rainer et al., 1998]. It has been suggested that dopaminergic cells projecting to the prefrontal cortex predict temporal difference (TD) error in primates and regulate the contents of working memory [Braver and Cohen, 2000]. These projections would allow the prefrontal cortex to select memories based on characteristics of other memories that have provided reward in the past. For example, if a driver involved in a hit and run accident did not remember the license plate of the other car, they would be punished by not receiving liability insurance money. The next time they were hit in such an accident, this information would be more likely to be retained as relevant to the task of fixing the car.

The TD-error is the difference between expected and actual reward discounted for the length of time in the future in which the reward is expected. The discounting dictates that a predicted reward in the distant future would be weighted less than the same reward in the immediate future. When the agent reaches the time at which it expects a reward but receives none, the error in expected reward is used to adjust the algorithm used for predicting future reward.

Since the TD-Learning algorithm used in reinforcement learning applications in computer science requires only reward information to learn useful knowledge about the environment, a TD-Learning-based system could provide a powerful, relatively automatic method for associating features in a system's environment with reward, therefore allowing the working memory system to select features resulting in higher reward. A generalized toolkit for selecting appropriate “chunks” has been developed using a TD-learning system

implemented with neural networks [Skubic et al., 2004], and it is planned to use this tool in Procedural Memory behavior generation as detailed in Chapter III [Ratanaswasd et al., 2005a].

In addition to learning to predict reward given by the environment to judge appropriate memory units for inclusion in the working memory contents, several other informational inputs could be used. For example, the system could judge a candidate chunk's context based on the other contents of the working memory system. Other information, such as an internally generated emotional salience signal or the novelty of a given memory chunk could be used to judge the most appropriate chunk for a given situation.

The ability to massively reduce the amount of information available to the cognitive system makes a working memory very useful for artificial cognitive systems. Relevant information is automatically presented to the cognitive system, allowing the system to concentrate on task execution instead of memory parsing and retrieval.

One criticism of the idea of a working memory as an active filter in an artificial cognitive system could be that the work of sorting through the environment for relevant information must still be done – the name of the process that performs this task is the only change with the introduction of a working memory system. This argument has two flaws.

First, extracting this element from the executive processes of the system allows for easy distribution of the work among multiple physical systems in the implementation of the cognitive system. The only data that need to be passed between cognitive processes and the working memory system are reward information and working memory contents, both of which are relatively low-bandwidth when compared to the massive amounts of

data required to sort through a large memory system looking for ideal chunks for placement in the working memory system. This also prevents the system from having to expend resources searching memory that could be spent planning or executing actions in pursuit of a goal.

Frequent memory searches are costly to the cognitive system, with traditional searches having time cost related to the size of the database by O(N) or O(log N). Although domain-specific heuristic search methods can be used to shorten this search process, the cost of searching a large database will not fall below that of searching a small, constantly-sized working memory system. If the working memory system were to have difficulty retrieving the specific element needed by the cognitive system, the system could then fall back to searching the raw database. Even this hybrid search strategy would drastically lower the cost of memory retrieval when compared to a system that had to search through a memory database for every desired element.

By extracting the working memory from the executive processes of the cognitive system, the executive processes can be designed with the assumption that the most relevant information is in the working memory. The ability to make this assertion dramatically simplifies the design of the cognitive system by reducing the amount of planning needed for correct memory retrieval.

The second flaw in the argument that a working memory system does not reduce the work performed by the cognitive system is that, by creating a specialized memory retrieval component, the retrieval of memory units may be maintained and learned over time. When thinking of an architecture that does not employ such a specialized memory retrieval system, the memory retrieval of the system would probably be performed by the generation of a new query and search every search request. Such a system would perform

similarly on the first and fiftieth searches. The separation of the working memory system, however, allows for the learning of characteristics of memory units that aid in retrieval, or, in the case of a generalized working memory system known as the Working Memory Toolkit, it allows for improvement of the results.

A program called the Working Memory Toolkit has been developed for the purpose of selecting arbitrary memory units with which to populate a working memory system [Phillips and Noelle, 2005]. The memory units and current state of the system are encoded into vectors. These vectors are then combined and compared with the use of a neural network that attempts to predict the future reward generated by selecting each candidate memory unit.

The toolkit is implemented in ANSI C++, and functions for encoding the system state and candidate memory units into vectors are supplied by the user. Because these functions are user-defined, the system is portable to a variety of working memory applications.

In the following sections, a strategy for the creation of a working memory system will be detailed. This system is heterogeneous because of the complex nature of the memory structures with which it operates, however so are many theories of human working memory [Baddeley, 1986].

ISAC: A Cognitive Robotic System

ISAC, or Intelligent Soft Arm Control, is a humanoid robot equipped with air-powered actuators designed to work safely with humans [Kawamura et al., 2000]. ISAC is equipped with two arms with position encoders for proprioception, two pan-tilt cameras for vision, IR proximity sensors for human detection, touch sensors on its hands,

a voice recognition system, and a sound-localization microphone array system. ISAC is able to interact with the outside world through arm movement, speech generation, and displays on its stomach screen.

ISAC was originally designed to safely feed disabled humans, but has been expanded to a research platform for human-humanoid robotic interaction and robotic embodied cognitive systems [Kawamura et al., 1995]. Figure 8 shows the ISAC robotic platform.

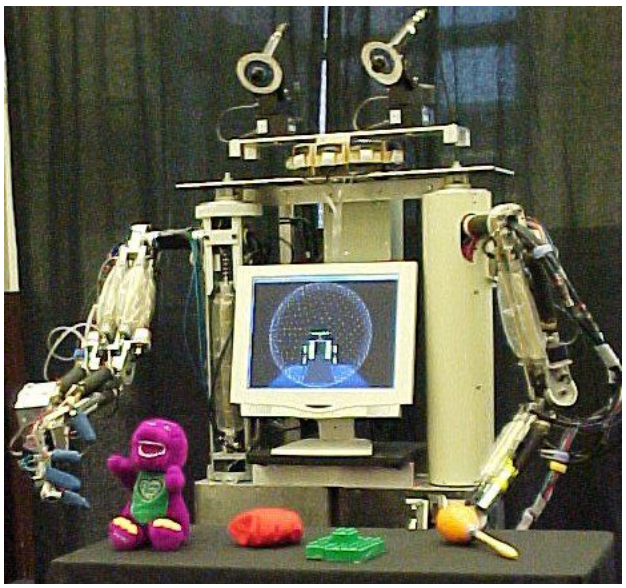

Figure 8. ISAC, a Humanoid Robot

Computation on ISAC takes place within the framework of the Intelligent Machine Architecture (IMA). IMA is a true multi-agent system designed to promote code-reuse [Alford, 1999]. Within IMA, “Atomic Agents” perform simple tasks, like Minsky's mental agents [Minsky, 1995], instead of containing goals, desires, and intentions like more complex agents [Franklin and Graesser, 1997]. In IMA, there are two “compound agents” that are aggregates of simple agents and function as a single agent:

the Human Agent and the Self Agent. Data is shared transparently between agents, and agents can be redesigned without changing the design of the rest of the system [Olivares, 2003].

IMA physically runs on multiple computers running the Windows operating system – multiple agents communicate using Microsoft's DCOM technology. Individual mechanisms that perform individual algorithms are programmed using C++ or Visual Basic, and then combined in a graphical environment to form agents. For example, the eye movement agent contains algorithms that both saccade to and smooth-track targets. These algorithms are combined within a single agent with the use of a state machine to change algorithms at appropriate times. Multiple agents are then aggregated to form ISAC's cognitive robot architecture.

ISAC's Cognitive Robot Architecture

ISAC's cognitive robot architecture is designed to learn to interact with humans in a complex environment without using explicitly programmed rules other than the initial knowledge needed to learn new rules. In other words, the system should be able to learn appropriate behavior through interaction with its environment without a programmer having to program the behavior (other than behavior needed for learning new behaviors). To accomplish this, a cognitive system has been created with the structure depicted in Figure 9 [Kawamura et al., 2004]

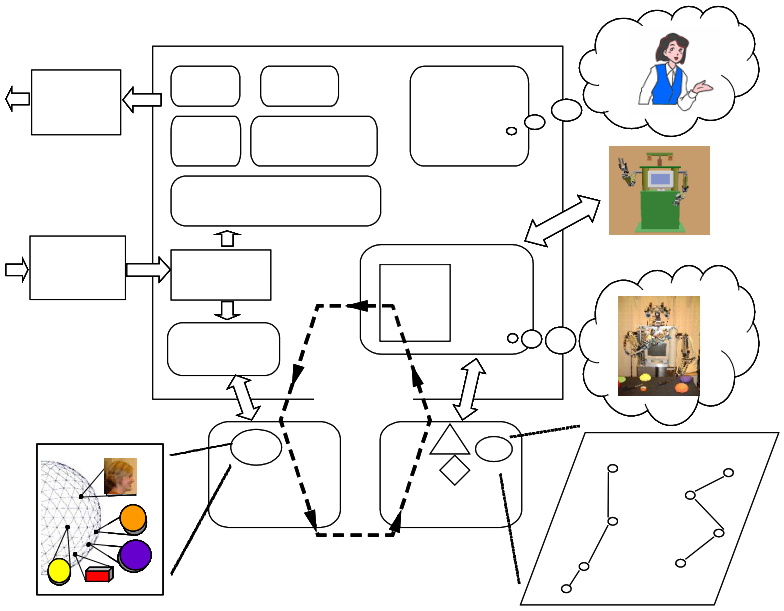

Figure 9: ISAC's Cognitive Robot Architecture

Figure 9 shows the components of ISAC's cognitive robot architecture. This system is embodied in ISAC – it interacts with the world through ISAC's actuators, voice, and computer display systems, and receives information about the world from ISAC's sensors.

ISAC's Perceptual System

Sensor input is processed by several perception agents. Each agent performs a single type of analysis such as color segmentation, voice localization and recognition, motion detection, or face recognition [Rojas, 2004]. Any number of agents can run simultaneously, posting their results to the SES Manager, the gateway to ISAC's short-term memory system. The SES is ISAC's short-term memory structure and is discussed in

more detail later in this chapter. Figure 10 shows the data-flow structure of the perceptual system.

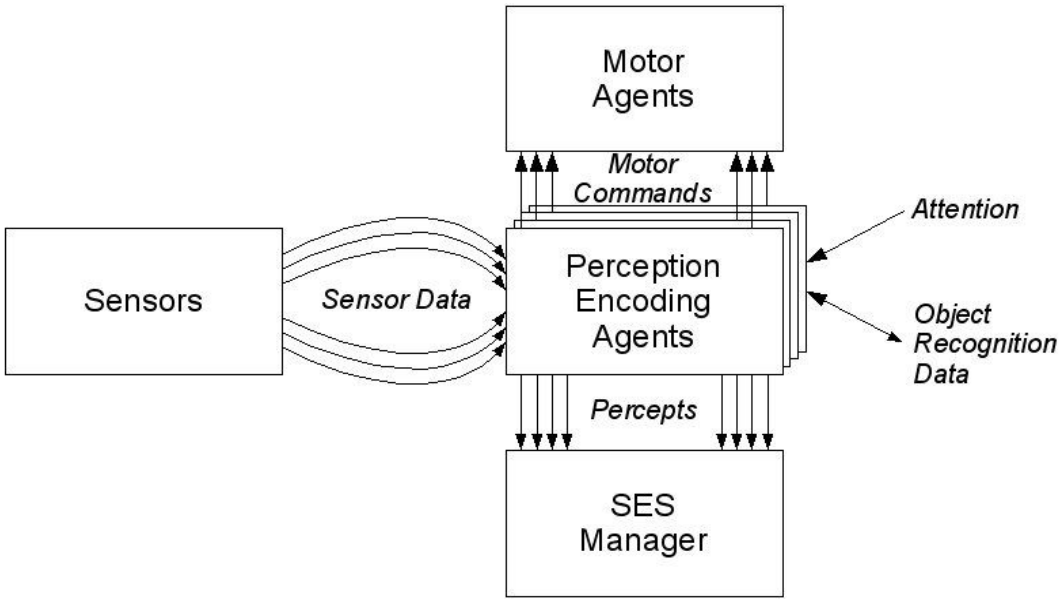

Figure 10. Perception System Detail

The agents can interface directly with the motor actuator agents, for both reflexive action and purposeful behavior such as scanning the environment or color following. The perception agents send percepts onward to cognitive elements of the system through the SES Manager. Perceptual elements can receive top-down attentional information from cognitive elements of the system, and can store and retrieve perceptual data from the Semantic Memory system (covered in Chapter IV).

ISAC's Memory Structure

ISAC's memory structure is divided into three classes: Short-Term Memory (STM), Long-Term Memory (LTM), and the Working Memory System (WMS). The

STM holds information about the current environment while the LTM holds learned behaviors (in the Procedural Memory or PM), semantic and perceptual knowledge (in the Semantic Memory or SM), and past experience (in the Episodic Memory or EM). The WMS holds task-specific STM and LTM information and streamlines the information flow to the cognitive processes during the task as detailed in the previous section.

ISAC's sense of self is represented in a compound agent (an agent consisting of several atomic agents) called the Self Agent. The Self Agent is the location of ISAC's planning system, executive control, self-monitoring, task selection, and other functions attributed to human cognition (see Chapter II).

A component of the Self Agent called the Central Executive Agent retrieves information from the memory systems through an active memory storage daemon called the Working Memory System. The Central Executive Agent is also responsible for executive functions such as setting tasks and goals, dealing with conflicting goals, and planning [Ratanaswasd et.al., 2005b].

The Working Memory System monitors the needs of the Central Executive Agent, the contents of the memory systems, and the output of the perceptual system to populate itself with the most relevant data pieces, or “chunks”. Each type of memory has a different method of chunk retrieval and its own limited-capacity Working Memory. Because the Working Memory is automatically populated with a limited number of chunks, the executive functions within the Central Executive Agent do not have to sort through massive amounts of information.

A structure called the Sensory EgoSphere (SES) holds STM data. The SES is a data structure inspired by the egosphere concept as defined by Albus [Albus, 1991] and serves as a spatio-temporal short-term memory for a robot [Kawamura et al., 2004]. The

SES is structured as a geodesic sphere that is centered at a robot's origin and is indexed by azimuth and elevation. In ISAC's case, the SES (as shown in Figure 11) is centered between the cameras.

The objective of the SES is to temporarily store percepts produced by the sensory processing modules operating on the robot. Each vertex of the geodesic sphere contains a database node detailing a detected stimulus at the corresponding angle.

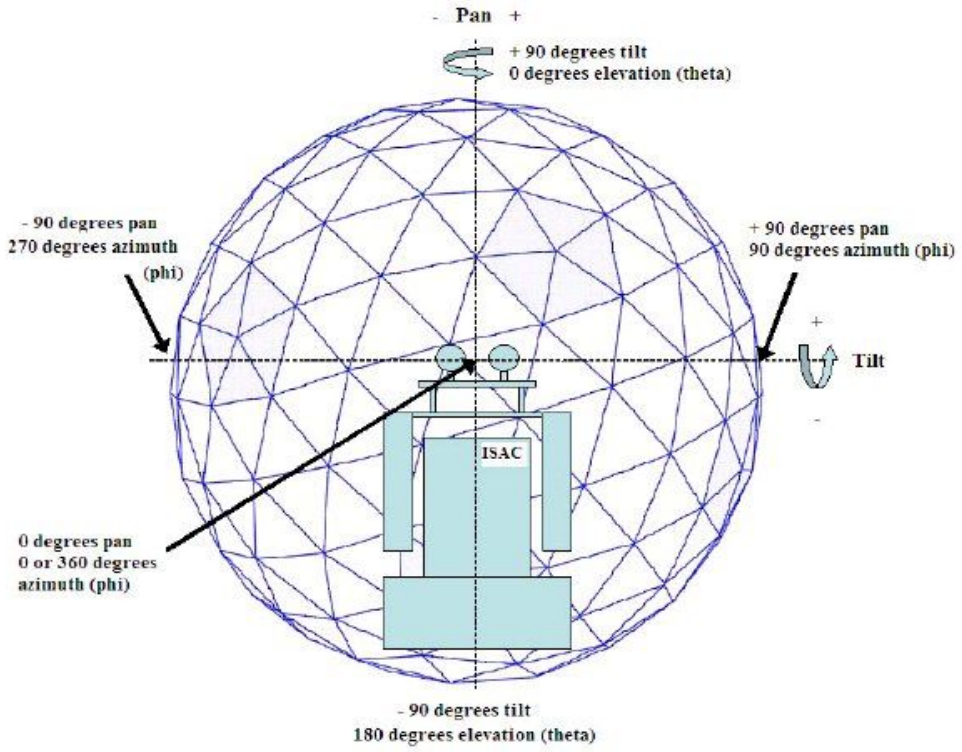

Figure 11. Structure of the Sensory EgoSphere [Achim, 2005].

Memories in the SES can be retrieved by angle, stimulus content, or time of posting. This flexibility in searching allows for easy memory management, posting, and retrieval. The memories in the SES also decay if they are not accessed, ensuring that only current and relevant data remains.

The Sensory EgoSphere can facilitate the direction of attention to external sensory events. Because sensory processors report all exteroceptive events to the SES, an attention network is able to search the SES for both task-relevant sensory data and unexpected yet salient sensory data and registers them to the ST-WM [Hambuchen, 1996].

As multiple events are registered in a common area, activation increases around a central node. Nodes that receive registration from task- or context-related events have their activations increased by the attention network. The attention network selects the node with the highest activation as the focus of attention. Sensory events that contributed to this activation are selected and those that fall within a specified time range of each other are passed into the WMS.

ISAC's LTM is divided into three types: Procedural Memory, Episodic Memory, and Semantic Memory. Like that in a human brain, the LTM stores information such as *skills learned* and *experiences gained* in the long term for future retrieval.

The part of the LTM called the Procedural Memory (PM) [Erol, 2003] holds motion primitives and behaviors needed for movement, such as how to *reach to a point*. Behaviors are derived using the spatio-temporal Isomap method proposed by Jenkins and Mataríc [2003].

To produce PM units, motion data are collected from the teleoperation of ISAC. The motion streams collected are then segmented into a set of motion primitives. The central idea in the derivation of behaviors from motion segments is to discover the spatio-temporal structure of a motion stream. This structure can be estimated by extending a nonlinear dimension reduction method called Isomap [Jenkins and Mataric, 2003] to handle motion data. Spatio-temporal Isomap dimension reduction, clustering and

interpolation methods are applied to the motion segments to produce Motion Primitives (Figure 12). Behaviors are formed by further application of the spatio-temporal Isomap method and linking Motion Primitives with transitions [Erol, 2003]. In other words, Behaviors consist of linked lists of Motion Primitives, which are small stereotypic motion segments.

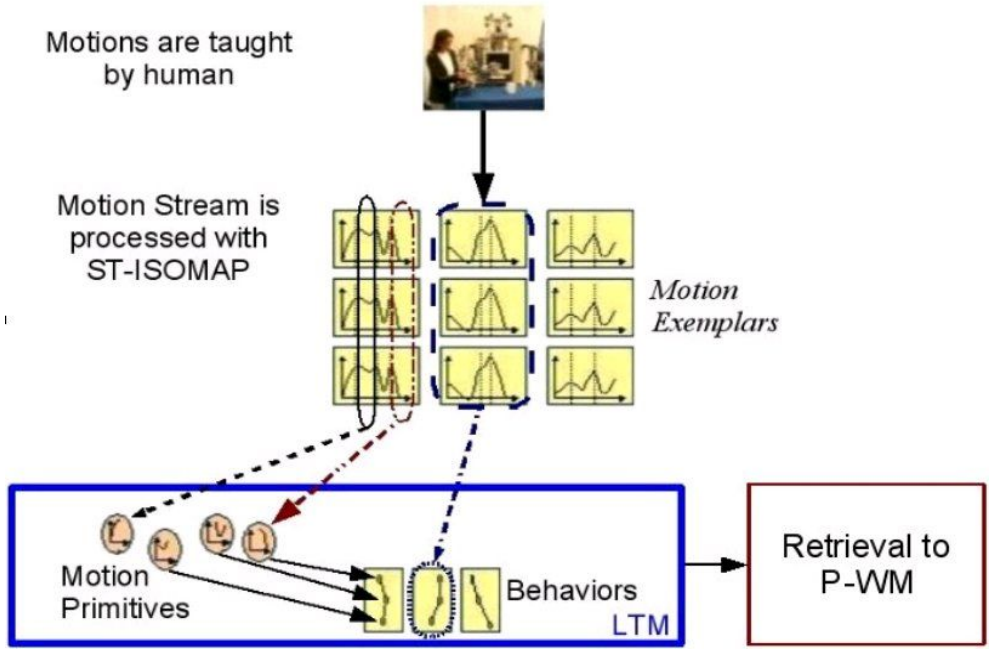

Figure 12. Derivation of Procedural Memory through a human-guided motion stream.

Motion skills for each behavior must be interpolated in order to be used in specific situations. In other words, the original recorded motion that produced the behavior must be altered to produce a desired motion to a different point. The interpolation method used is the Verbs and Adverbs method developed in [Rose et al., 1998]. This technique describes a motion (verb) in terms of its parameters (adverbs) that allow ISAC to generate a new movement based on the similarity of stored motions. These

values are used to interpolate a new motion after a behavior is deemed relevant and retrieved to the P-WM.

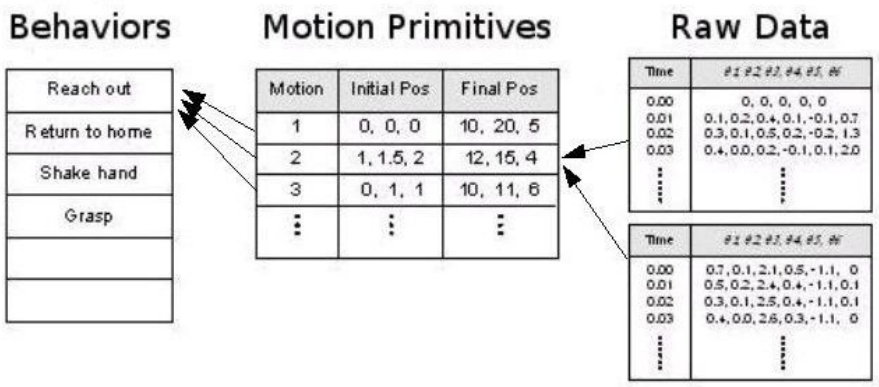

Figure 13. Structure of Procedural Memory data unit.

Figure 13 depicts a current representation of the PM data structure. At the top of this structure, behavior descriptions will be stored allowing the system to identify what each behavior can contribute to solving a given motor task. Each entry in the behavior table contains pointers to the underlying motion primitives, which are linked to the actual segmented data constituting that particular motion primitive. The memory formation and retrieval process is described in more detail in Chapter III.

The Episodic Memory (EM) system holds records of past experiences in a time-indexed format. The Episodic Memory daemon records the contents of the Working Memory System along with reward information from the emotion system (for more details on ISAC's emotion system design, see Chapter V) for the duration of a single task and posts this information to an EM data unit. EM units decay at a rate related to the output of the emotion agent for the duration of the recorded task, and are retrieved using a method inspired by human memory performance (See Chapter V).

The Semantic Memory (SM) is a data structure containing information about objects in the environment. It details perceptual and semantic information, and is intended to allow the cognitive systems to reason about and plan with objects as well as to allow for the combination of sensor information to form a single concept of an object through the perceptual processes. In other words, a single semantic memory unit should represent a fire alarm recognized by both beeping noise and flashing light, even though the recognition process involved in interpreting these two stimuli is probably performed by separate perceptual agents.

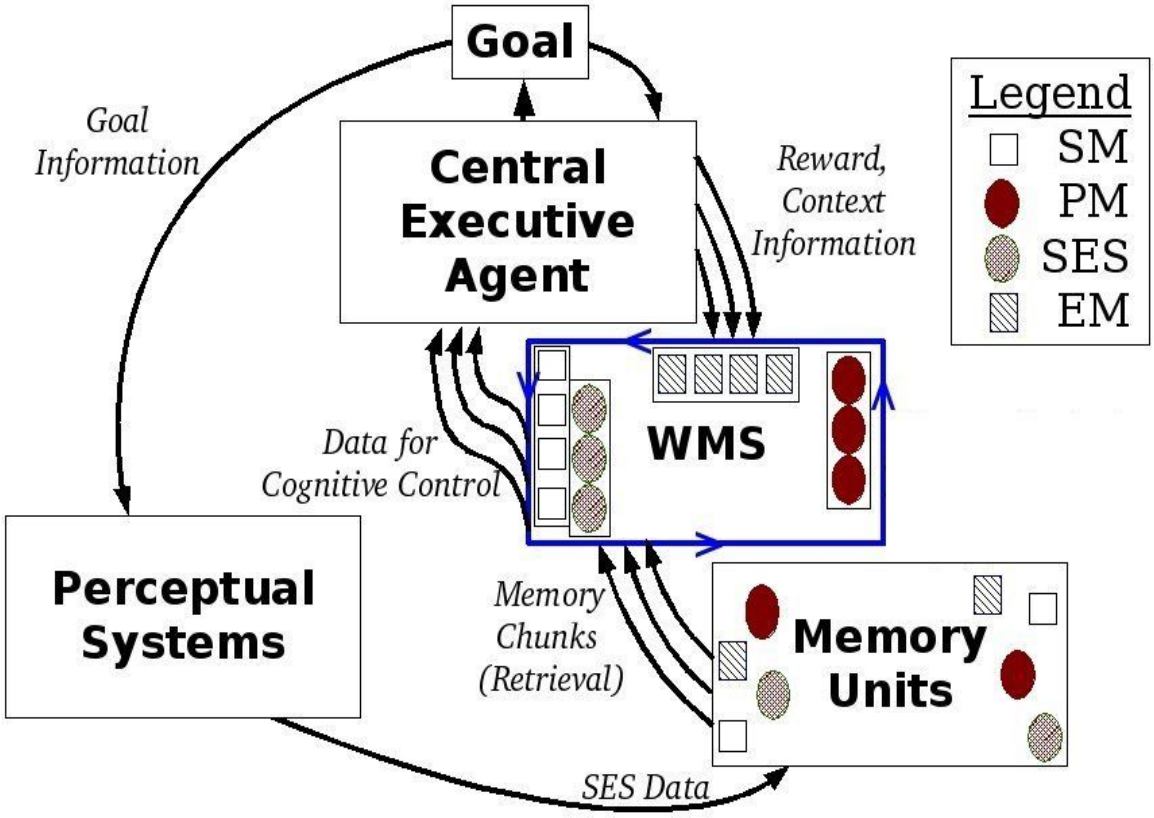

Figure 14. Logical Function of the Working Memory System.

Figure 14 shows the function of various parts of the WMS. The multiple lines represent links where multiple memory units are actively maintained and transferred, while the single lines represent the feedback between the CEA and memory systems through perceptual direction and attention. The arrows surrounding the WMS are meant to show that it is an adaptive, changing system that adapts to the contents and situation of both the memory banks and the CEA.

The design of the LT-WM takes many different forms. The S-WM holds the Semantic Memory information needed to complete the task and is implemented by a simple keyword search as described in Chapter IV. The E-WM is populated by an agent that selects EM units based on similarity to the current situation and emotional salience. For example, if ISAC were attempting to place the red bag in the box but had never attempted this task before, the E-WM would retrieve episodes in which ISAC had performed similar tasks or tasks within the same environment. EM units themselves are generated by the contents of the whole WMS during task execution. The EM generation and E-WM population algorithms are described in more detail in Chapter V. The P-WM holds behaviors needed to complete the current task, and serves to reduce the complexity of real-time error-driven execution of behaviors. These behaviors are selected through a reinforcement learning algorithm that attempts to predict each behavior's use to the current movement task as discussed in Chapter III. Each memory type is tested and analyzed with techniques outlined in Chapter VI.

The Compound Agents

ISAC's cognitive architecture contains two compound agents: the Self Agent and the Human Agent. The Human Agent handles human interaction, while the Self Agent is intended to represent ISAC's sense of self [Kawamura et al., 2005].

The Human Agent (HA) is responsible for ISAC's human interaction [Rogers, 2003]. The HA detects, tracks, identifies, models, parses the speech of, deciphers the gestures of, and speaks and gestures to humans. The Human Agent communicates directly with the Self Agent, allowing humans to interact closely with ISAC's cognitive system.

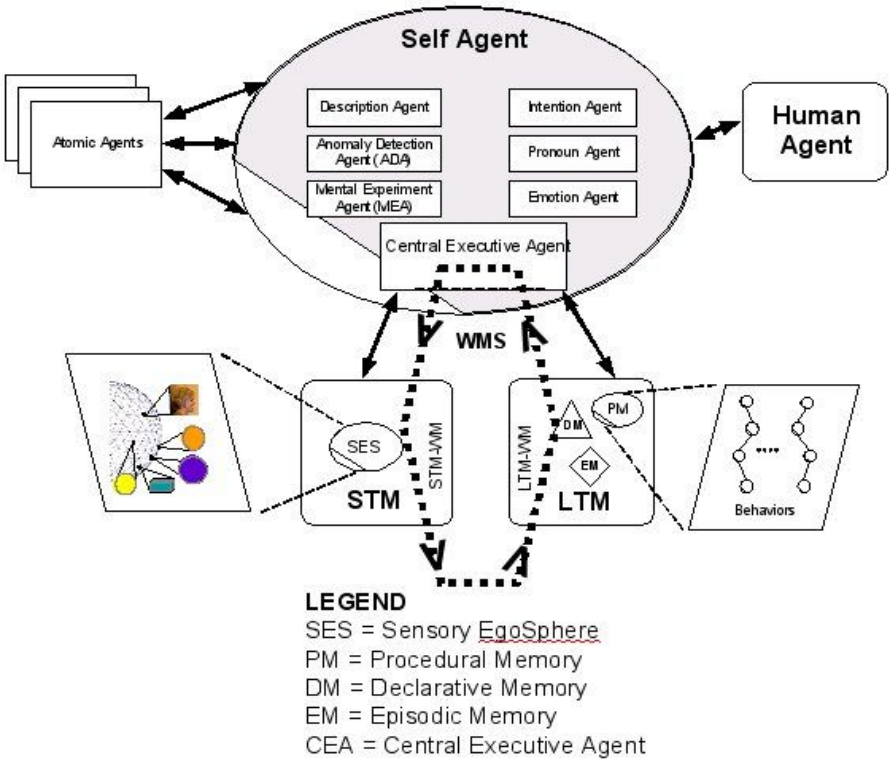

Figure 15. The Structure of the Self Agent [Kawamura et al., 2005]

Figure 15 shows the structure of the Self Agent (SA). The SA represents ISAC's sense of self by monitoring task execution, system health, and the robot's internal state.

Currently, the Description Agent, the Intention Agent, and the Pronoun Agent have been implemented. The Central Executive Agent and the Emotion Agent are under development [Dodd et al., 2005] [Ratanaswasd et al., 2005b]. The Emotion Agent is of paramount importance to the correct operation of the Episodic Memory, as is discussed in Chapter V.

The emotion agent was simulated for the memory experiments because of its incomplete state. Chapter VII, however, discusses how the Episodic Memory system may be tuned to resolve any inconsistencies between the outputs of the actual implementation of the Emotion Agent and its simulation.

CHAPTER III

PROCEDURAL MEMORY AND PROCEDURAL WORKING MEMORY

The Procedural Memory (PM) holds information needed for the generation of ISAC's movements [Erol, 2003]. The system is best understood by following a single movement, from the teaching process to movement execution by the robot. The contribution made by this thesis is the design of the database node to hold and retrieve the information output by the following process.

First, ISAC's arm is moved along the path of the behavior. The arm can either be moved by a person standing next to ISAC and moving the arm, or by a computer controller issuing commands to move the arm. The movement trajectory is called a behavior exemplar, and forms the base information of the higher-level behavior data structure. Note that the output of the control system can be recorded to expand the behavior library, allowing for multiple procedural memories to be combined into a single behavior for future applications. This sort of memory re-encoding and combination may be seen in the human brain [Fuster, 2000].

The behavior exemplar is fed through the ST-ISOMAP system, which combines several exemplars of the same behavior and breaks the combined information into several temporal sections, or Motion Primitives (MPs). For example, a 20 second reaching motion might have a 5 second segment where the shoulder moves, followed by a 7 second segment of quick motion by the elbow. These two segments would be MPs.

The Procedural Memory system stores the output of the ST-ISOMAP system in three database nodes, each nesting several of the next: Behaviors, MPs, and

ExampleNodes. Currently the ST-ISOMAP system outputs a text file containing the name of the behavior, the location of several behavior examplar files, and the positions of the MPs within the exemplar files. This data is parsed and encoded in the correct structures.

The ST-ISOMAP algorithm segments data at boundaries where characteristics of the data change dramatically. The segments between these boundaries constitute the MPs, which are stored in the data structures described in the following section.

Procedural Memory Storage Implementation

The Procedural Memory system is stored in the cross-platform open-source MySQL database system. All memory storage and retrieval scripts are written in the Perl language, and the modular controller detailed at the end of this chapter is written in C++.

The PM data unit is encoded into a linked database in the structure of Figure 16. The arrows denote links to data nodes. There are three types of nodes in the database, from highest level of abstraction to lowest: Behavior Nodes, MP Nodes, and Exemplar Nodes.

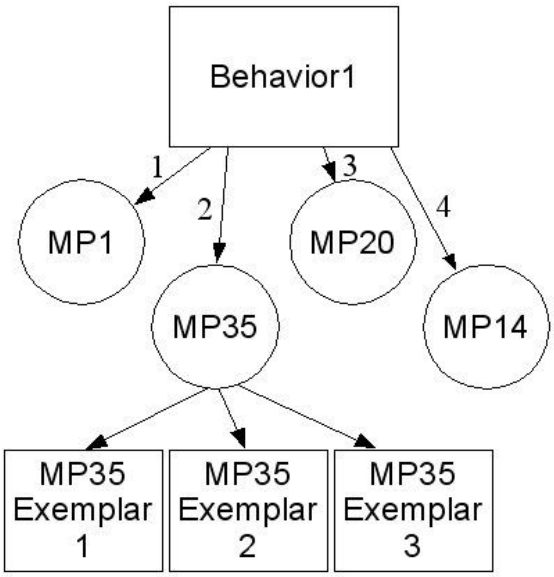

Figure 16: PM Database Structure

The highest level of the PM database hierarchy is the behavior node. This is the node that is retrieved when interpolating a new motion for action execution. This node may be seen in Figure 16, and is labeled “Behavior1”.

Table 1. PM Behavior Node Structure

Field Name	***Description***	***Example Data***
id	Unique identifier, used for retrieval and tracking	Behavior 1
name	Set by user, used for feedback to humans	Reach out
lengthInMPs	Length of Behavior in Motion Primitives	3
MPs	Forward Links to MP nodes	MP 32, MP 24, MP 53
timestamp	Datestamp of Behavior formation and access	3:30 PM, July 2, 2005
description	Used for system debugging and to allow for contextual information to be associated with memory units for future implementation with the Working Memory Toolkit	<blank>

Table 1 shows the structure of the Behavior node. The "MPs" field contains links to all the Motion Primitive nodes that contribute to the particular behavior. As is shown in Figure 16, the links are an ordered list of all MP nodes constituting a single behavior. The timestamp field is updated each time the behavior is updated or accessed, and can be used for memory pruning and consolidation. "Description" is not currently used, but can hold textual descriptions of the behavior or information needed for converting the behavior to chunks for the working memory system.

Table 2. PM Motion Primitive Node Structure

Field Name	***Description***	***Example Data***
id	Unique identifier for MP, used when referencing MP from Behavior node	MP 1
RawData	Link to id of Exemplar nodes associated with the MP	Exemplar 24
ReverseDeps	Behavior nodes containing links to this MP, used for housekeeping	Behavior 2, Behavior 5
Statistics	Holds semantic information for future memory consolidation	<blank>

Table 2 describes the implementation of the PM node. In Figure 16 the MP data nodes are circular. The "RawData" field holds links to Examplar Nodes containing sample data for this particular Motion Primitive. These links are in no particular order, and multiple examples may be specified for each Motion Primitive. The "Statistics" field could be used for more intelligent consolidation (consolidation by similarity), or to hold statistical information for converting behaviors to chunks in the working memory system. "ReverseDeps" holds reverse links to all behaviors that are partially composed by a

particular Motion Primitive, and is used to find orphan nodes when deleting behaviors from the memory database.

Table 3. PM Exemplar Node Structure

Field Name	***Description***	***Example Data***
id	Unique identifier	Exemplar 3
data	Raw data constituting MP units	<joint data>
InstanceOf	Reverse links to MP nodes linking to this node, used for housekeeping	MP 2

Table 3 details the implementation of the Exemplar Node structure. In Figure 16, the Exemplar Nodes are the rectangles on the bottom and hold the raw data in Motion Primitive number 35. “data” holds the raw data found in the examples, while “InstanceOf” holds the id number of the Motion Primitive node of which the Exemplar node is an instance. The “InstanceOf” field is used for housekeeping and consolidation.

For all three nodes, forward and reverse-links are maintained both for housekeeping and data retrieval. All links refer to a node's “id”, which is configured using MySQL's Primary Key functionality to achieve minimal search time when retrieving a node.

Procedural Memory Decay

Removal of Behavior Nodes is performed by timestamp-pruning. Behaviors that are not retrieved frequently are logically the ones that the system does not use. The Modular Controller, the system that uses the Procedural Memory units, currently contains a reinforcement learning system that attempts to evaluate the utility of each of the PM units in memory. Because the reinforcement learning system evaluates these units, units

that are unlikely to be used by any motion command may be found by calculating the set of all behaviors that will not be retrieved by any motion command (motion commands are discussed in Chapter IV). This pruning is important for correct operation of the Modular Controller system as discussed in Chapter VII.

Procedural Memory Retrieval & System Integration

Robust control of motion is essential for robotic applications in which the robot must complete tasks and learn from experience within a complex environment. It is of no use to plan a complex solution to a task that the robot is incapable of executing, just as it is useful to be able to automatically learn which behaviors are appropriate based on the context of the task.

The contribution made to the following system through this thesis is the design of the algorithm that selects PM units for use in the modular controller. More details about this system may be found in [Ratanaswasd et al., 2005a].

ISAC's Modular Controller is a system which learns both which behaviors to select before motion execution and the best path to take during motion execution. To do this, it combines the concept of working memory with behavior interpolation and error-driven execution [Ratanaswasd et al., 2005a]. Figure 17 shows how the system combines existing behaviors (shown in solid lines) to interpolate a new movement (shown by the dotted line).

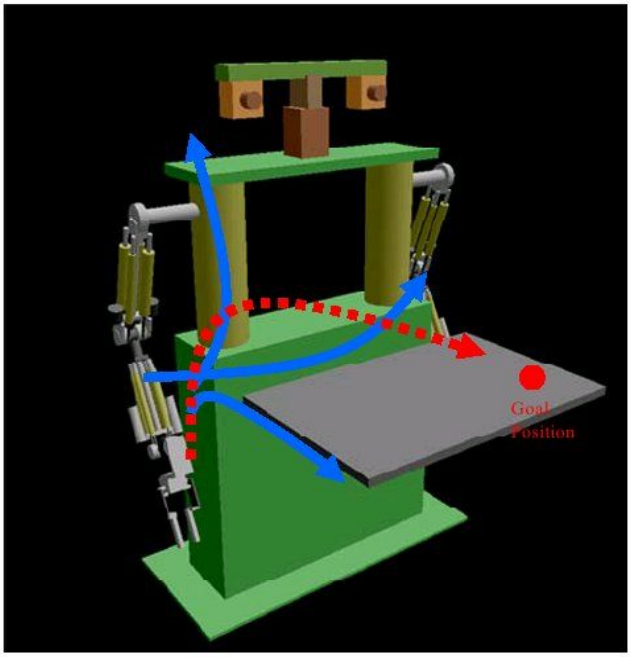

Figure 17. Behaviors are Combined for ISAC's Movement [Ratanaswasd et al., 2005a]

The mechanism used to accomplish this interpolation across behaviors is called the Modular Controller. Figure 18 shows the structure of the modular controller from a functional standpoint.

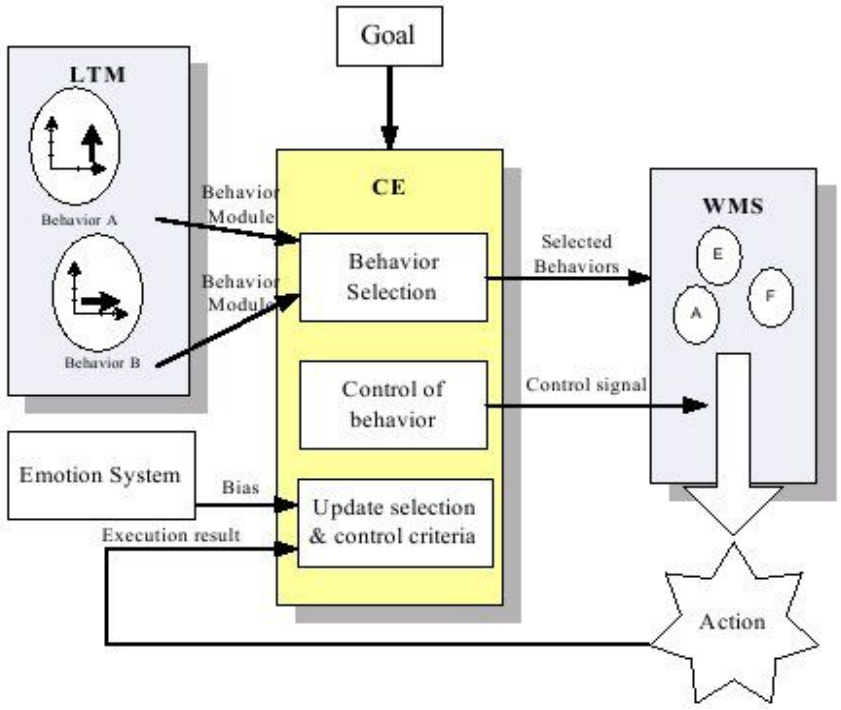

Figure 18. Modular Controller Functional Structure [Ratanaswasd et al., 2005a]

Figure 18 shows the functional structure of the Modular Controller. Behaviors are held within the Procedural Memory which is a section of ISAC's Long-Term Memory. These behaviors are selected by a reinforcement learning process within the Central Executive Agent that evaluates the expected reward provided by a given behavior. The selected behaviors are stored within a structure called the Procedural – Working Memory and are then combined in an error-driven manner to produce the desired behavior. After a movement is generated, reward is calculated for the set of selected behaviors so the system can better predict how a given behavior set will function in the future.

Figure 19 shows the implementation of the modular controller. The mechanisms that select the PM units for placement in the P-WM are contained within the Central Executive Agent. After interpolation, each behavior contains the information needed to estimate the current state within the context of that behavior and to predict the next state

produced if that behavior is followed. In other words, when the motion is executed, each behavior contains information needed to determine the weighting of the behaviors at the next step in behavior execution.

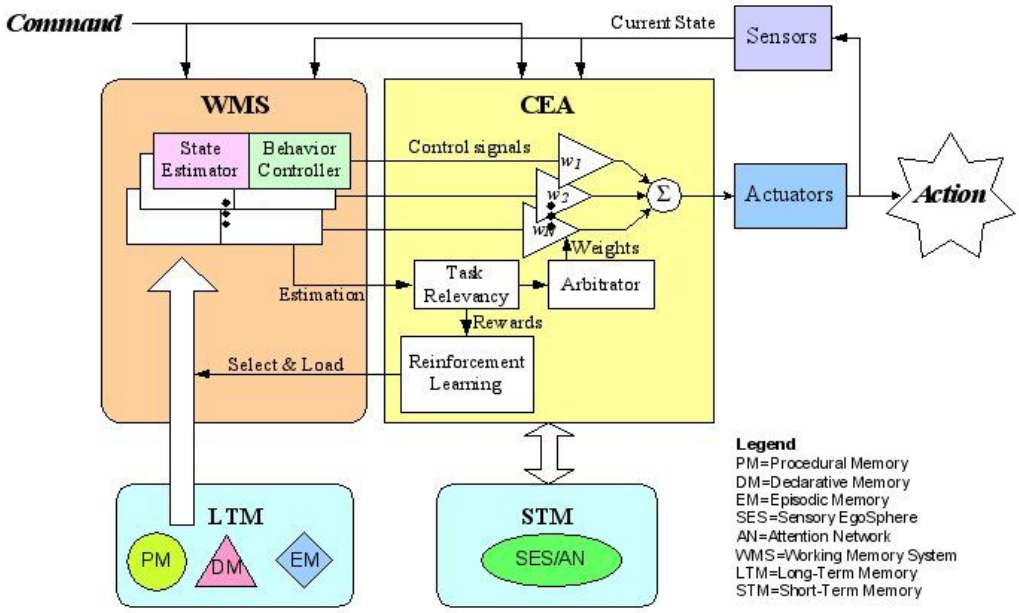

Figure 19. Implementation of the Modular Controller [Ratanaswasd et al., 2005a]

It may be seen in Figure 19 that behaviors are weighted and summed in order to produce motion. The formula that the Central Executive Agent uses to compute these weights may be found in [Ratanaswasd et al., 2005a]. The contribution of this thesis is the system that selects behaviors for inclusion in the P-WM and adjusts predicted reward based on the performance of those behaviors.

The Procedural Memory – Working Memory System (P-WM) holds a set of three behaviors that are to be used to accomplish the movement task. The working memory size was set to three to allow the system to achieve a high sampling time when executing behaviors in an error driven manner while retaining the benefits of combining several

behaviors to produce novel motions. The calculations that find the relevancy of each behavior for the next motion execution time step are fairly complex, requiring a limited number of behaviors for which these calculations need be made. By selecting this limited number of behaviors for placement in the P-WM, the time required to make these calculations is reduced and the sampling time for movement execution may be raised.

These behaviors are currently selected by a reinforcement-learning technique based on TD-Learning. The behaviors are selected before the movement execution time, and reward is supplied after the movement is executed to provide the selection system with feedback.

The role of the reinforcement learning system is to select the behaviors that will be appropriate for the current movement task. A new reward is calculated for each set of behaviors after attempting the task. The reward is equal to the average relevancies (or weights) of behaviors within the set over the time it took to complete the task, discounted by the task completion time. For more information on the specifics of these calculations, see [Ratanaswasd et al., 2005a]. In this manner the system attempts to select behaviors that are both fast and useful for a given task. No reward is given if the system fails to achieve the goal state.

Thus, the set of behaviors that complete the task most quickly with the most precision will be rewarded and are likely to be selected in the future. ISAC learns which set of behaviors to select for each task. New, similar tasks can be built on the learned values.

Currently the reinforcement learning system is implemented by a simple look-up table that attempts to estimate the utility of each PM unit and is used to select the top three behaviors for the task. This selection system is also configured to conduct a random

exploration for a small percentage of task attempts in order to prevent initially explored modules from "hogging" activation, and to guarantee that the system will converge on a usable, reasonably explored state. This approach has drawbacks, however, as detailed in Chapter VII. Changes to this behavior selection process are also described.

This system attempts to estimate the reward for each behavior in the system, with a simple weighted averaging technique used to adapt the values stored in the estimator to match the utility function of the behaviors to the Modular Controller and movement task. In the future, the working memory toolkit [Phillips and Noelle, 2005] will be integrated into the system to replace the current, table based system. This neural-network based system will be more robust than the current selection mechanism to changes in task context, as is discussed in Chapter VIII.

After the behaviors are loaded into the P-WM, each one is interpolated as close as it can to the destination location using the Verbs and Adverbs interpolation technique [Rose et al., 1998]. The motion is then executed, combining the behaviors with different relevancies during execution to reduce error in the trajectory path (for more information on this process see [Ratanaswasd et al., 2005a]). Finally, reward information is calculated to evaluate the PM chunk selection using the formula discussed above.

The Procedural – Working Memory prevents the system from having to interpolate a trajectory for each behavior in memory. It also significantly lessens the resources needed for real-time error-driven control by reducing the number of behaviors to be combined. The savings produced by this reduction change the cost of executing a motion from hundreds of calculations per movement time-step to three relevancy calculations. This intuitively results in smoother motion generation with many available behaviors because the sampling time for behavior generation is increased.

CHAPTER IV

SEMANTIC MEMORY AND THE SEMANTIC WORKING MEMORY

The Semantic Memory (SM) currently holds information about objects in ISAC's environment. It holds both information for perceptual systems on how to recognize individual objects and information for cognitive systems that can be used for reasoning about these objects. The Semantic Memory can not be queried directly from the CEA, but is queried through links to SM units contained within Episodic and Short-Term Memory structures.

Figure 20 shows the Semantic Memory information flow from sensors to the CEA.

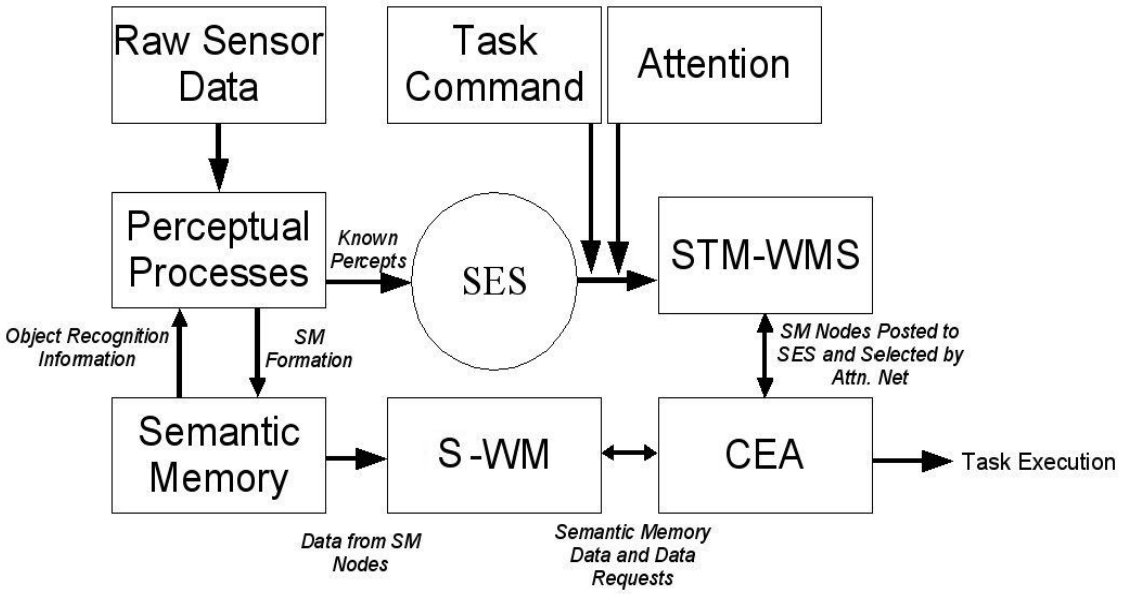

Figure 20. Semantic Memory Information Flow Through Perception

Figure 20 shows how information from the perceptual system of ISAC travels to the Central Executive Agent (CEA). Links to SM nodes are communicated through the Short Term Memory (STM) and the STM-Working Memory System as is shown in the top path of Figure 20. The second way that links to SM units are communicated to the CEA is through the Episodic Memory, which records the contents of the ST-WM and the S-WM. For clarity, Figure 20 does not detail this process. It is, however, discussed in Chapter V.

The method through which the CEA can retrieve more detailed knowledge of a Semantic Memory node is through direct query from the CEA to the Semantic Memory Working Memory System. Within this system, the CEA requests SM nodes about which it needs more information. For example, the CEA could request the retrieval of memory nodes from the S-WM after retrieving an episode from the Episodic Memory that contained a crucial SM unit. Note that the CEA is not permitted to search for memory units within the S-WM, but can only look for elements referenced in the SES or an Episodic Memory unit.

Figure 20 shows the pathway through which percepts (links to SM nodes) make their way to the CEA. When an object is perceived, a link to that object's SM unit is posted to the SES. If that node is selected to pass through the attentional network and into the ST-WM, the CEA has access to the SM contents that denote the object and its physical location..

Semantic Memory Encoding

Memories in the Semantic Memory are directly generated by the perceptual system. A sample perceptual system that generates SM nodes is detailed in Chapter VI.

There must be some initial knowledge that is encoded into the Semantic Memory system before ISAC's cognitive system can function. Such initial knowledge should include information on how to recognize objects and how to learn what the identity of objects are in order to build new knowledge about objects. For example, the ability for a perceptual system to detect novel objects is useless if ISAC's system can not understand a human telling it the identity of the new objects.

Perceptual agents should be created that impose some initial knowledge into ISAC's SM system, as well, without explicitly endowing ISAC with specific SM units. For example, a perceptual agent that specifically identifies the location of a human's pointing finger would be useful for many human interaction tasks in which ISAC might participate, however it is the creation of the agent running a given algorithm that allows the system to identify this crucial knowledge, not a preexisting SM unit.

Another example would be an agent that can identify brightly colored objects. The initial knowledge contained within this agent would be the ability to identify the objects, but the SM information generated by this agent would be specific to observations made.

Figure 21 details the hierarchy of the Semantic Memory system and shows how the “Barney” object would be represented within this structure. The main SM node is the Object node. This node is referenced within the Short-Term Memory and Episodic Memory systems, and forms the atomic unit of information that is passed around within these systems.

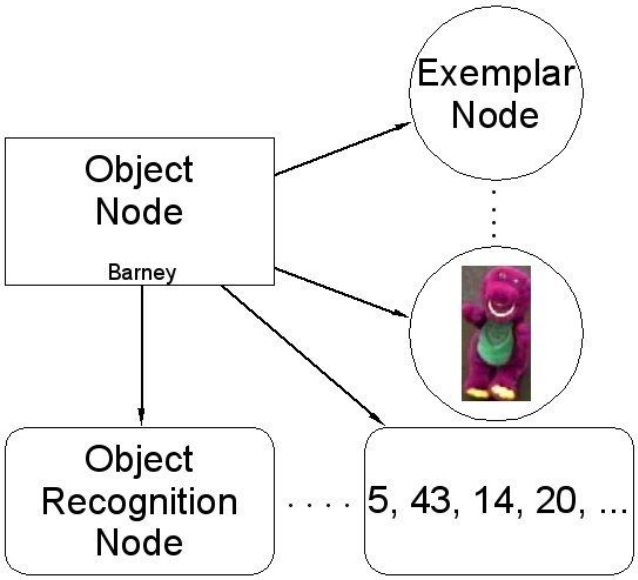

Figure 21. Structure of the Barney SM Node

Figure 21 shows the structure of a specific SM node representing the Barney doll object in ISAC's environment. The numbers in the Object Recognition Node in the lower right side of this diagram represent the recognition coefficients for a given perceptual process, while the picture in the Exemplar Node on the right side of the image represents the sensor data that caused the perceptual process to identify the stimulus as Barney.

The Object node is created by a perceptual process, and represents a distinct (known or novel) percept that the perceptual process identified within the environment. After the perceptual process identifies a novel object, it creates a corresponding object recognition node that holds all information that the perceptual process needs to identify that object in the future.

A single object may have multiple object recognition nodes in order to allow multiple perceptual processes to use the same SM node, representing the target object in a more robust manner. For example, the Object node in Figure 21 represents the Barney

doll. This node would have been created by a perceptual process that identified a novel characteristic in its particular modality.

The perceptual process would have then created a new node, and stored the information that it needed to identify the object (the Object Recognition node) and the sensor data that prompted the identification in the first place (the Exemplar node). If the perceptual process recognized objects based on color, it would have stored the numbers representing the color purple in the Object Recognition node, and a small picture of Barney in the Exemplar node as seen in Figure 21.

The Exemplar node holds the raw sensor data that was translated to the percept represented by the Object node by perceptual processes. The Exemplar node is used for such purposes as: training new perceptual algorithms on old data without having to show each object in turn, displaying objects in GUI applications for user feedback (i.e. the SES display program), and allowing the system to interact with humans more naturally (for example, by asking a human what a particular image is if ISAC didn't recognize an object identified by a novelty detector in the perceptual agent).

These applications are possible because the recorded sensor data is the same as sensor data that is perceived in real time. Training an algorithm or displaying this data can be done from memory the same as if it were performed using current sensor data.

For example, a tone identification agent, upon identifying a novel sound frequency, would generate an Object Recognition node containing the frequency of the new tone. The Exemplar node would contain a recording of the tone.

Semantic Memory Consolidation and Decay

Semantic Memory units should decay when they have no use to either the cognitive system or to perceptual agents. This can be performed through a pruning of the Object Recognition node particular to a given perceptual agent upon the loss of relevance, followed by a general database sweep pruning Object nodes that do not reference Object Recognition nodes.

Although this approach requires the cognitive system to generate an Object Recognition node when it finds a given Semantic Memory unit useful, this pruning may be achieved through a single database command.

Semantic Memory Retrieval

SM Memory units are retrieved through separate methods for perceptual and cognitive processes. Perceptual processes retrieve SM information through direct search of the Recognition nodes for relevant object recognition information. The perceptual processes need to be able to look for a wide variety of objects in the environment, and have access to the entire database of Semantic Memory Nodes. Search methods are left to the perceptual agents.

The cognitive processes retrieve data through the S-WM, a memory store that holds the several most relevant Semantic Memory nodes currently being used. This working memory is populated by queries for single memories that have pointers contained within the active Episodic Memory nodes.

Semantic Memory Implementation

The Semantic Memory system is implemented as a location and platform-independent MySQL database [Dubois, 2003]. Active components of the system, such as the components that prune unreferenced SM nodes, are implemented as scripts, interpreted text files that run like programs. All scripts are implemented in the similarly platform-independent Perl language, allowing the memory systems to run in a distributed manner.

Perl was selected due to its compatibility with the MySQL database, as well as its power to parse and translate between complex data types of different formats.

Table 4. Object SM Node Structure

Field Name	***Description***	***Example Data***
id	Unique identifier	Object 1
Name	String identifier of object	Barney
Type	Type of object	Doll
RecInfo	Links to RecInfo nodes containing information needed by perceptual processes to identify the object	RecInfo 3, RecInfo 5
SampleData	Links to SampleData node containing the raw sensor data from which the object was recognized	SampleData 2
Characteristics	Holds semantic information used for human feedback and future reasoning	Soft, plush, brightly colored
Refs	Number of Episodic Memory units referencing this Object (See Chapter V)	4
EpisodeLinks	Links to those Episodic Memory units	EM 3, EM 5, EM 7

Table 4 details the structure of the top-level SM node, the Object node. This node holds semantic information in the “Characteristics” section, as well as links to the input

data that prompted the recognition of the object (SampleData node) and the recognition information specific to the perceptual algorithm used to identify the object and create the Object Node (RecInfo node). The "Characteristics" section can hold any text, and is intended to hold supplementary information needed for planning or human feedback about specific objects. This information will be set by the CEA, and allows the SM system to be extended in the future. The "Refs" value denotes the number of Episodic Memory nodes that reference the Object structure, and is used for finding SM nodes that are safe to delete because they are not referenced by any EM units.

Table 5. RecInfo SM Node Structure

Field Name	***Description***	***Example Data***
id	Unique identifier	RecInfo 14
InstanceOf	Links to object that references this node	Object 3
Method	Contains name of the perceptual agent that uses this node	RGB Color Segmentation
Data	Information needed to identify object (used by perceptual algorithms)	R=3, G=60, B=128

The RecInfo node, as seen in Table 5, holds information on how to recognize objects. This node is linked to by the RecInfo field in the Object table shown in Table 4. Multiple RecInfo nodes may be linked to within a single Object node in order to allow for multiple sensor modalities to identify a single object.

The RecInfo node is implementation dependent, any coefficients or recognition information can be placed in the "Data" field, and a particular perceptual engine running

a recognition algorithm can query the database for nodes using a particular recognition method for its own use.

For example, the colorOvuloid perceptual system (as discussed in Chapter VIII) could query the database for all colorOvuloid Data fields containing a certain value.

Table 6. Example SM Node Structure

Field Name	***Description***	***Example Data***
id	Unique identifier	Example 3
Modality	Sensor from which data was drawn	Right Camera
Data	Sensor data from which object was identified	<image of Barney>
InstanceOf	Reverse link to Object node referencing this node	Object 1

Table 6 shows the Example SM Node structure. This node holds an example of the stimulus recorded from the stated modality. For example, it would hold a picture of the Barney for Barney recognition or a recording of Dr. Kawamura's voice for recognizing Dr. Kawamura through voice. It is linked to in the SampleData field of the Object Node seen in Table 4. The Data field holds the raw sensor data, while the Modality field holds the sensor modality that provided the data.

Semantic Memory Decay

The Semantic Memory consolidation and decay system is implemented by a simple script that prunes Object nodes not linked to any Object Recognition nodes.

Semantic Memory Retrieval

Because retrieval of Semantic Memory units for perception is performed directly by the perceptual processes, there are no explicit scripts that handle the retrieval of SM for the purpose of perception.

Each perceptual process should directly search the database in a manner appropriate for its own purposes. The "Method" field of the RecInfo node as seen in Table 5 allows perceptual processes to retrieve all relevant recognition information with a single query, and to search through the "Data" fields within this set of nodes to find the appropriate recognition information.

The purpose of the S-WM is to hold information needed for task execution by the Central Executive Agent. The Central Executive Agent should specify which SM units about which it needs more information, and should populate the S-WM through direct query. As such, it does not require an independent script for S-WM population.

CHAPTER V

EPISODIC MEMORY AND THE EPISODIC WORKING MEMORY

The purpose of an Episodic Memory system is to allow the learning and representation of episodes, or temporally sequenced records of specific events that occurred to the cognitive agent [Nuxoll and Laird, 2004] [Tulving, 1985]. Within the computational neuroscience community there is a theory that there are two mutually exclusive goals of long-term memory: learning interleaved, generalized representations of behaviors and perceptions and learning isolated, one-shot episodes [McClelland et al., 1995]. The former memory formation type is generally attributed to densely activated neuronal firing patterns, or codes, within the human neocortex, while the latter is achieved through sparse coding in the human hippocampus [Norman, 2003].

An effectively designed episodic memory system must accomplish the task of retrieving the correct episodes for a given situation. In the following section, we will outline how ISAC's episodic memory system is designed to store and retrieve the relevant memory using memory context and emotion.

Episodic Memory: Storage, Retrieval, and Decay

As stated earlier, the EM system holds records of specific, temporally-based past experiences, or episodes. An episode is defined as a period of task execution of the robot during which the goal of the robot does not change. The Episodic Memory (EM) is intended to provide cognitive processes with the ability to evaluate and learn from past task performances. It holds a record of the contents of the SM and ST-WM as well as

task-related information from the Self Agent for the duration of the task. This data is stored alongside the output from the Emotion Agent, which is used to determine the decay of the episode.

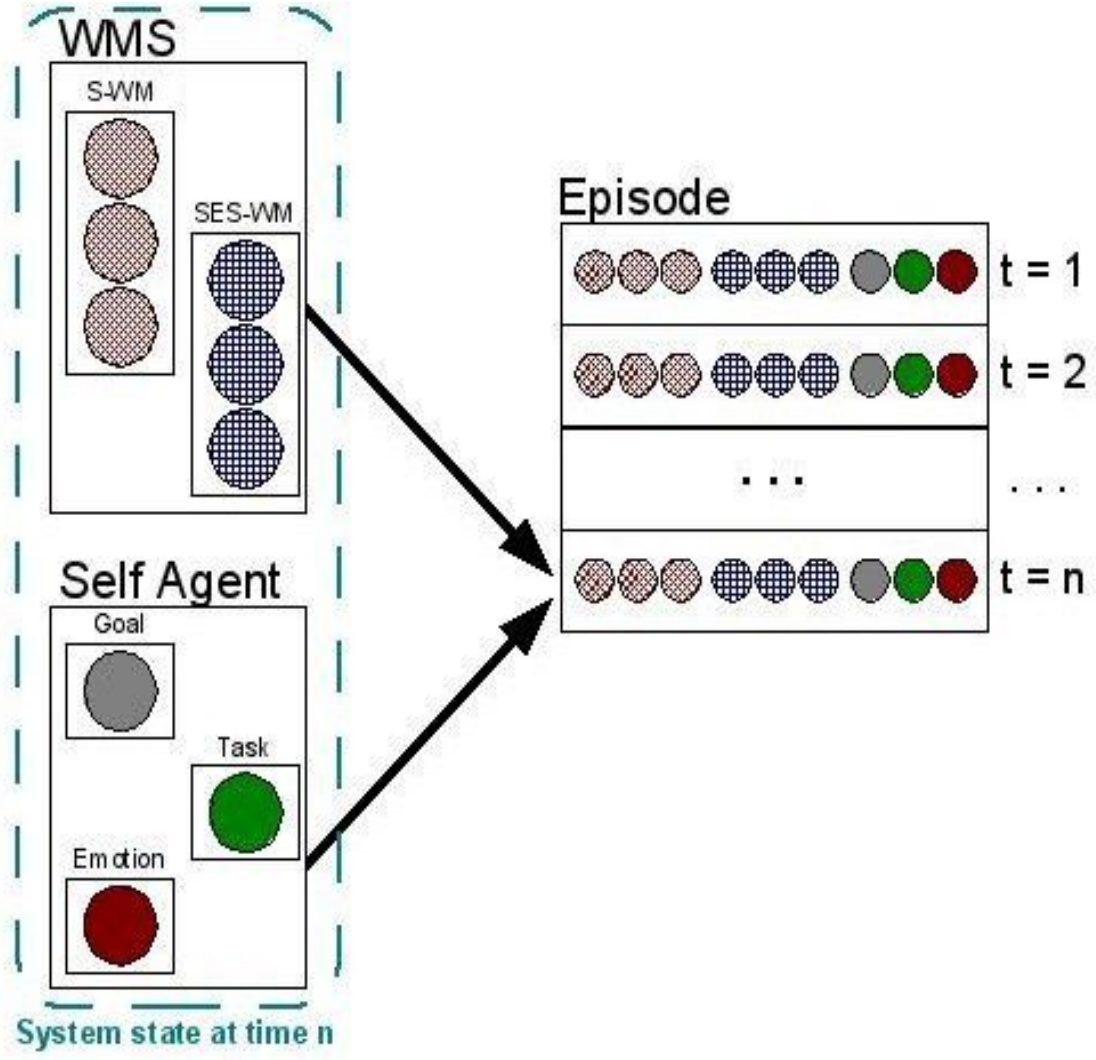

Figure 22. Episodic Memory Formation [Dodd et al., 2005]

As seen in Figure 22, each episode contains links to the entire contents of the Working Memory System and Self Agent as well as the output of the Emotion Agent for

the duration of the task. All constituent links point to Semantic Memory units, so statistics may be calculated about the frequency of use for each SM.

The task of the EM retrieval system is to automatically populate the Episodic Memory Working Memory System (E-WM) with the correct episodes for a given situation. The correct episode is the one that provides the information needed by the executive system for planning or task execution. Because the correct episode could provide the location of a lost object, a plan for a task to be used in case-based reasoning, or even a clue as to why some percept in the environment is associated with such strong emotion that it warrants a goal change, the "correctness" of an episode is very difficult to judge.

The relevance measurement for comparing a given episode with the current situation is defined as consisting of five major criteria that serve to highlight several scenarios in which an episodic memory unit would be used.

- The retrieved episode should contain common SM units to the current situation.
- Recent, commonly accessed episodes are more useful than older, less used episodes.
- Novel similar percepts or goals should score higher than common similar percepts or goals within episodes.
- The CEA should be able to shift attention to desired aspects of a target episode.
- Emotionally salient memories should score higher than those producing little emotion.

The first two criteria are obvious: if the cue (the forming episode) contains a specific task or object, the resulting memory needs to contain common elements to be

relevant to the current situation, and recently formed memories are more likely to contain applicable information.

The third criterion is used to enhance the score of cues that are rare over those that are commonly seen. This serves to enhance those aspects of a cue that differentiate it from others. For example, if ISAC's environment contained a red bean bag with which it interacted many times a day and a black box with which it had interacted only once before, it would be beneficial to retrieve the episode detailing the context of the box for a possible goal change.

The fourth criterion allows the Central Executive Agent to weight aspects of a cue that are important to the current context. For example, if the system needed to know the last place it placed a particular object, it would need to heavily weight the memory structure representing that object.

Lastly, a machine emotion system [Haikonen, 2003] may be used to interject personal significance to an episode. If the robot were strongly rewarded after performing a task in a certain manner, it makes sense to recall that episode over a task execution in which little feedback was given. This emotional salience allows for personal reward calculation as well as reward calculation for

The method in which EM memories are retrieved is based on the work of John Anderson [Anderson, 1990]. In this book, Anderson demonstrated that a rational approach to memory retrieval applies to many domains, from human memory to library databases. This approach demonstrates that the “rational” approach to memory design is that which maximizes the benefits of considering a memory unit for retrieval considering the cost of this consideration. Having to consider a large number of memories to find the

desired one, therefore, would be strongly discouraged. The first equation used for ISAC's Episodic Memory retrieval is the cost equation [Anderson, 1990]:

$$pG \geq C \tag{5.1}$$

This equation states that the probability that a given memory unit is the correct memory unit times the reward of reaching the goal should always be greater than or equal to the cost of retrieval. This characteristic is implemented in the Episodic Memory system through the use of a threshold for acceptable retrieval success. In other words, the system operates in a fuzzy manner to retrieve those elements that are likely to be the correct ones and not to waste resources considering memory units that are unlikely to be useful. It is also implemented by pruning nodes that can not exceed the cost of retrieval.

The retrieval of episodes is accomplished through an algorithm that takes the currently forming episode and selects several stored episodes for placement in the E-WM. As (more generally) stated by Anderson [Anderson, 1990], the probability that a memory is relevant is calculated through the combination of two independent factors: a history component P(A|H) and a contextual component P(A|Q) [Anderson, 1990].

$$P(A) = P\left(A|H_A\right) * P(A|Q) \tag{5.2}$$

The contextual component P(A|Q), or the probability that the memory denoted as A is suitable given only the cue denoted as Q, is calculated by measuring the importance of each constituent SM unit in the formative EM and comparing it with the relative importance of the same SM in the candidate EM units. This is similar to comparing

common elements of a collection of data to determine the importance of individual elements. A memory histogram is computed by taking the percentage of time that each SM spends in the Episodic Memory divided by the the total size of that memory. The histogram is then altered to reflect the influence of contextual information from the CEA and frequency of occurrence information about the constituent SM unit. This process will be described in more detail shortly (Figure 25 shows this process in detail).

The history component, $P(A \mid H_A)$, is insensitive to the current context. It is determined by the emotional salience and age of the episode. The rationale behind this design is that, if an episode contains high emotional salience, it is likely to be important to the cognitive agent. For example, if ISAC were punished the last time it ignored a certain percept, it would be beneficial to retrieve the episode of that punishment the next time that stimulus was encountered.

The retrieved episodes will be used to generate future actions through a planning system within the CEA. Because the size of the E-WM is relatively small, episodes can be linked together to chain behaviors in a tractable fashion, even if brute force methods are applied. The E-WM is populated continuously, so the episodes available to the planner change as new situations arise. This allows the system to generate plans as new problems are encountered.

The process for generating plans could be considered a form of case-based reasoning [Leake, 1996]. This approach could provide the system with a way to learn from experience and generate plans without having to generate a complicated statistical model to generalize over multiple experiences.

EM units can also be used for the extraction of information linking a particular SM to a particular time and place. An example of this type of extraction is a person

remembering the location of the car they parked an hour previously, or when they last spoke with a friend.

The search that populates the E-WM is performed in a top-down manner – the list of SM units in the search cue (created by calculating the fingerprint of the forming episode) are ordered by importance as shown in Figure 25. EM units containing each SM unit are found through the use of a hash, and each candidate EM unit is assigned a score by multiplying the importance of the SM unit to the candidate unit with the importance of the SM unit to the cue. The list of SM units in the cue is traversed in order, and candidate EM units are pruned as it is no longer possible for them to enter the list. When the list of candidate units equals the size of the E-WM, the search is successful because it is not possible for any other episodes to enter the list. The search is analyzed in Chapter VII, and psuedocode of the search process is given later in this chapter.

This search method has several characteristics that are advantageous to its application in ISAC's cognitive system:

- It prunes irrelevant information from memory during the course of a search, reducing memory requirements.
- It weights less common constituent SM units more than others, so the search space is naturally constrained.
- It requires only a simple hash, reducing resources needed for keeping track of complex data structures.
- The search will return the best possible results at any given time during execution because the search traverses the fingerprint of the forming episode in descending order.

The history component of Equation 5.2, or the probability that memory A is appropriate given its history, is calculated by the equation [Anderson, 1990]:

$$P(A|H_A) = \frac{v+n}{M(t)+b} * r(t) \tag{5.3}$$

In Equation 5.3, v and b are constants and shape the decay curve and are set to 1 and 3 by [Anderson, 1990] although they can be altered to tune the memory system, M(t) represents the integral of the decay function r(t) (defined in equation 5.4), and n is the number of times that particular EM unit has been accessed and brought into the E-WM.

Equation 5.4 shows the decay function of the EM unit, where α represents the emotional salience of the EM unit (adapted from [Anderson, 1990]).

$$r(t) = \alpha e^{-\alpha * t} \tag{5.4}$$

Figure 23 shows sample decay curves for several emotional saliences. Those memories stored with lower alpha coefficients do not decay as quickly as those with higher alpha. The alpha is therefore set to be inversely proportional to the salience of a given memory term. The salience term is important for the selection of correct episodic memory units – the robot should strongly retrieve episodes that resulted in extreme reward or punishment, and it should remember those highly salient episodes well into the future.

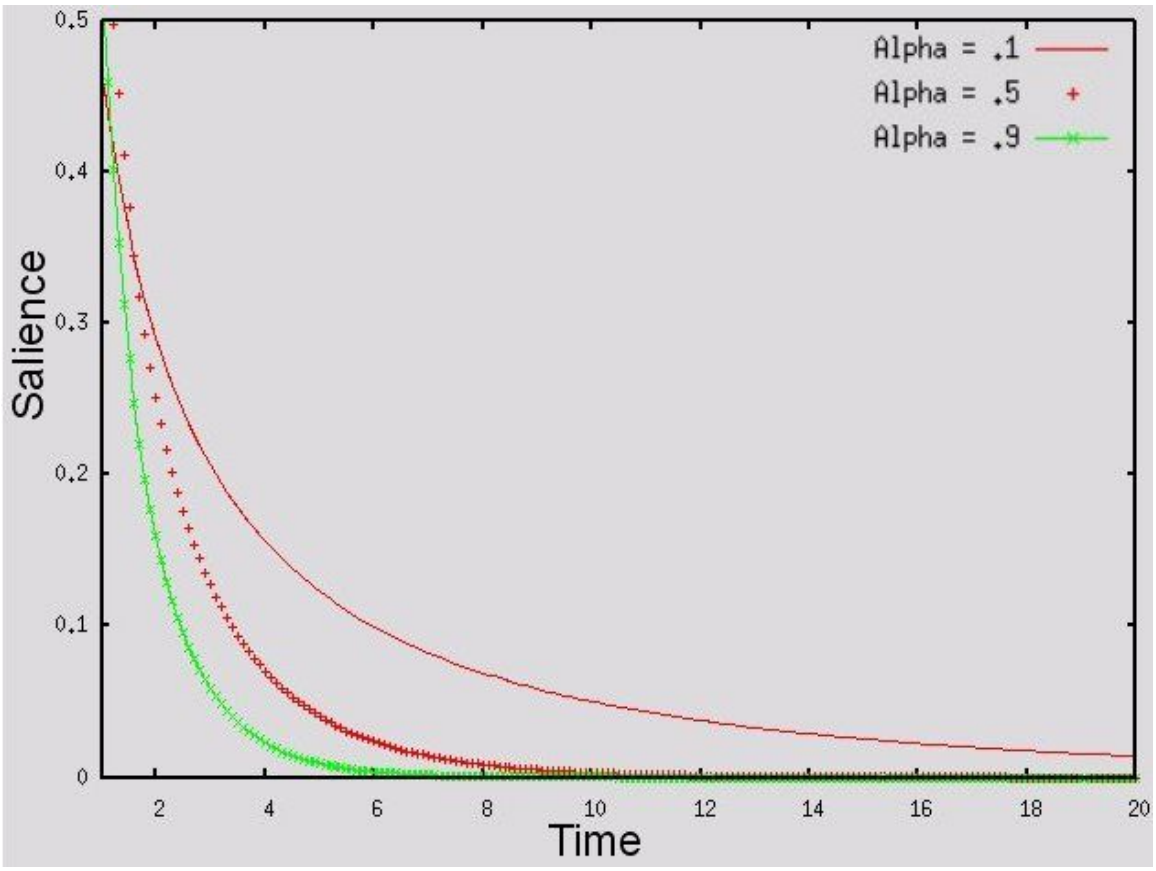

Figure 23. Sample decay curves for various alpha

The plots in Figure 23 show the decay of memories that are not accessed. When a memory is accessed, its history probability increases because of “n” in Equation 5.3. This helps offset the decay for a memory that proves to be useful in several situations.

Humans may be seen to exhibit this quality of remembering emotionally salient events as well in what are called “flashbulb memories” [Conway, 1995]. For example, a person will remember (or have the impression of remembering) the moment when they heard traumatic or extremely good news. In ISAC, the salience of a memory is calculated through a transformation of the emotional salience scalar that is the output from the Emotion Agent during the formation of the episodic memory. The generation of this scalar is discussed more in the next section.

The history component of this value is independent of the current context, and therefore may be computed independently. A memory is removed from the system when its history value decays past a cutoff point because the probability that the memory will be relevant to the current situation given its history value is less than the approximated cost of considering it for retrieval. Every memory that is considered in the search causes the space and time complexity of the search to increase, causing the performance of the cognitive system (which must either wait for results or cut the search off early and settle for inaccurate results) to drop.

ISAC's Emotion System

ISAC's Emotion system interjects personal meaning to the Episodes and is used to calculate the salience of a given Episodic Memory. The emotions are computed using a method based on that proposed by Haikonen [Haikonen, 2003].

It is important to note that ISAC's emotion system is currently under development. This section details the view proposed in [Dodd et al., 2005] for generating emotional scalar values, and is included in this thesis to clarify the Episodic Memory search process.

Figure 24 shows the structure of ISAC's Emotion System.

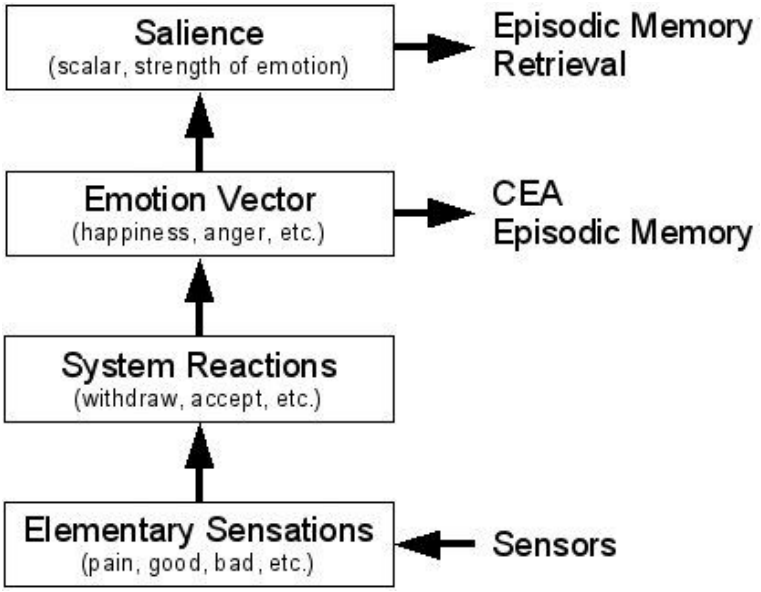

Figure 24. Structure of ISAC's Emotion System [Dodd et al., 2005]

The lowest level of ISAC's emotion system is Elementary Sensation. This could be such a thing as associating bright light shone in the cameras as bad or extension of a joint to its limit as pain. Another elementary sensation might be that the successful completion of a task is good. It is important to note that not all sensors need be external sensors; some sensors could be situated within the CEA to provide feedback on the state of the system. These elementary sensations are then mapped to system reactions, that provide ISAC with a base impetus on which to derive actions.

The system reactions are then combined to form a larger emotion vector representing the emotional state of the whole system through a mapping function that links system reactions with their corresponding emotions. This emotional signal is sent to the CEA and is recorded within the Episodic Memory in order to provide ISAC's emotional state to future task executions.

The emotion vector is condensed to form a measure of the emotional salience of the current system state through a measurement of magnitude. This is a scalar and is used

to generate the alpha decay values of Episodic Memory units. Because the decay speed of an episode is inversely related to the strength of its emotional salience, the alpha value is calculated from the salience s (which has a range of [0,1]) by Equation 5.5.

$$\alpha = 1 - s \tag{5.5}$$

This causes Episodic Memories that are emotionally salient to decay much slower than those that do not evoke strong emotion.

It is important to note that ISAC's Emotion System is not currently implemented, so all EM retrieval results are derived from simulation of this system. This simulation was conducted through direct generation of salience values as listed in Appendix B. Because the mapping from emotion vector to salience value is formulated to provide the Episodic Memory with the correct saliences, it may be tweaked to produce the desired values for a given situation.

Implementation of Episodic Memory

The EM itself is contained within a MySQL database; each EM unit is a flat data structure containing all the information detailed in Figure 22.

The context value, meanwhile, relies solely on the contents of the current memory unit. In order to negate temporal issues regarding the comparison of different segments of an EM unit or different temporal sections of a single task, each EM unit is reduced to a "fingerprint" consisting of its constituent SM units and how much they each add to the EM unit as a whole.

The term "fingerprint" is used to denote the characteristics of the data structure representing the importance of the constituent SM units within the forming episode. The current system state and context of the state can be characterized by this data structure.

The occurrence of a given SM unit x may be defined as:

$$\alpha_x = c(x)/n \tag{5.6}$$

In Equation 5.6, α_x is the xth element of the occurrence vector, n is the total number of SM units in the episode, and c(x) is the count of SM unit x in that particular EM unit. The α_x vector is therefore a measurement of the frequency of SM units within an EM.

The W vector for each SM unit X is defined as:

$$W_x = B / c(SM) \tag{5.7}$$

In Equation 5.6, B is a weight that can be used for placing more emphasis on important SM units like goals and human commands. The B values are adjusted to fit the retrieval needs of the cognitive system. For example, if the system is looking for a particular object, attention should be directed toward that object, while if the system is trying to develop a plan to accomplish a task, the system should be tuned to retrieve based more on task similarity. c(SM) is the total occurrence of the SM unit, and is a measure of how common it is within other EM units. In other words, it is a reference count. If c(SM) = 0, W is set to equal 0 for that SM unit and a signal is sent to the cognitive system that

the detected SM unit is novel to the episodic memory and should perhaps be provided with semantic knowledge.

The alpha vector is multiplied elementwise by the weight vector, to produce a weighted "fingerprint" vector F. This vector consists of tuples containing the SM id of the constituent parts as well as the weighted fingerprint values. The fingerprint is then normalized and the vector is sorted by fingerprint values. We will call this vector "L". The entire process is illustrated in Figure 25.

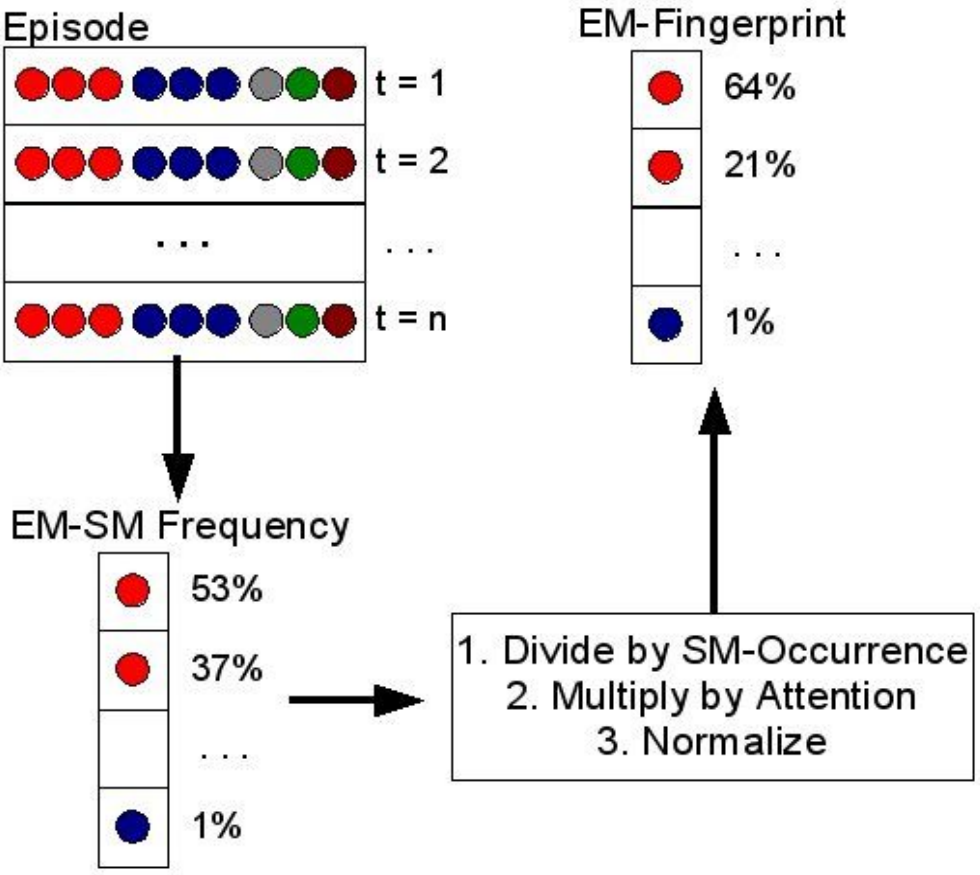

Figure 25. Fingerprint Formation Process

It is important to note that the retrieval process uses the EM-SM Frequency table for candidate EM episodes. This table may be pre-calculated during memory encoding and formation.

The retrieval algorithm follows:

1. Take L_0, the first element of the sorted L vector, and look up the hash corresponding to that particular SM unit.
2. Calculate scores for each of the EM units in the hash by multiplying the fingerprint value for the SM unit in question within that EM unit, the history factor of that EM unit, and L_0.
3. Push scores from EM units in the hash onto a result vector R, summing scores of the same EM units if they are already on R.
4. Sort R by score.
5. For threshold t equal to the next element of L – tolerance where tolerance represents the degree to which the search may be cut short to conserve time, discard all values x in R where $R_{wmsize} - R_x > t$.
6. Set t = t/[EM History Value] for each unit and repeat step 5 to remove any episodes that cannot move up in the list any more than they already are.
7. If size(R) = wmsize, return all EM pointers from R as the WM contents.
8. Otherwise, shift L and return to step 1.

Episodic Memory Pruning Implementation

Memory pruning is closely tied with retrieval. When a memory is not reasonably considered for retrieval due to decay, it is discarded to keep the database clean. This is accomplished through the use of a decay process running in the background that deletes a given EM node when it decays beyond a certain point. The point should be derived experimentally based on the performance of the system in the real world.

This may be justified by Equation 5.1. The cost of retrieving a memory must be balanced with the probability that the memory will be useful. Memories that will probably not be useful may be discarded.

CHAPTER VI

EXPERIMENTAL DESIGN

The experiments detailed in this chapter are intended to analyze the future performance of ISAC's cognitive system by testing each of the three memory systems (Procedural, Semantic, and Episodic) individually. The Procedural Memory experiment differs from the experiments for the other two because it involves output of action from the CEA and the interaction of the CEA with the P-WM. The other two memory systems are tested to ensure that the creation and WMS retrieval processes operate without interaction with the CEA. These memory experiments and analysis are intended to demonstrate how the memory systems will fit into ISAC's cognitive system as a whole. Results and discussion are presented in Chapter VII.

Procedural Memory Experiment

The PM experiment is meant to determine the efficacy of the Procedural Memory system in generating movement behaviors to accomplish a certain task. A library of behaviors is to be generated to demonstrate the PM encoding process, and the correct operation of the memory encoding and retrieval systems are shown.

Procedural Memory Experiment 1

The first experiment is designed to show the correct operation of the modular controller. A set of five behaviors is generated by moving the arm in a sample motion and recording the motion. The number five is chosen to ensure that none of the behaviors

were redundant, and none could interpolate to the target point. These behaviors are manually loaded into the Procedural Memory system, and are segmented into Motion Primitive exemplar segments using a MATLAB implementation of ST-ISOMAP. Then, the behaviors are stored in the Procedural Memory as discussed in Chapter III.

The Modular Controller (Chapter III) is then signaled to reach to a point that was not able to be reached to through the use of any single behavior. This point is chosen to ensure that the system has to combine behaviors in order to generate the desired motion. Correct behaviors are selected based on each one's estimated reward, and motions to that point are interpolated using the Verbs and Adverbs technique. It is important to note that the Reinforcement Learning system starts in an essentially random state, and requires a period of time in which it learns which behaviors provide good reward. After selection, the motion is executed using the method discussed in the modular control section of Chapter III.

Error is calculated after the motion is executed, and the performance of the Working Memory System at generating the desired motion is analyzed over several runs to show how it is learning. The experiment will be run on the PISAC simulator, a simulation of ISAC's hardware that allows for the testing of control algorithms [Ratanaswasd et al., 2005a]. The simulator also provides an error-free environment for testing the control, behavior selection, and reward calculation algorithms (described in Chapter III) without the necessity of designing filters to smooth the system input.

This experiment therefore tests two main functions of the Procedural Memory: encoding and retrieval. If the experiment can be performed at all, the encoding system is judged to be functional because the system is using the behaviors stored within the PM to

generate a motion. If the rate of successes rises over time, the retrieval process in which chunks are selected for the working memory system is judged to be working correctly.

Procedural Memory Experiment 2

The second PM experiment is intended to test the response of the retrieval mechanisms to an increase in the database size. The first experiment will be repeated with two additional behaviors added to the library. The added behaviors are designed to be useless to the generated motion, so the robustness of the performance of the selection system to additional candidate behaviors is measured. Both behaviors involve cyclic movement at waist-height, while the target point is set near the upper chest. The robustness of the behavior selection system is estimated through measurement of the decrease in performance.

Semantic Memory Experiment

The Semantic Memory experiment demonstrates the correct implementation of a perceptual agent that operates with the Semantic Memory system outlined in Chapter IV. The system will be shown to recognize objects in different locations with different recognition coefficients (recognition coefficients are generated by the perceptual algorithm being demonstrated), and the recognition linked to in the Semantic Memory Object nodes will be shown to evolve as more exemplars are recognized. This provides the perceptual system with a way to generalize over multiple perceptions of a given object.

Semantic Memory Experiment 1

This experiment will test the operation of the retrieval of Semantic Memory units by perceptual processes for object identification. It will also show the creation of new SM units for novel percepts. Figure 26 shows the components that are tested by this experiment.

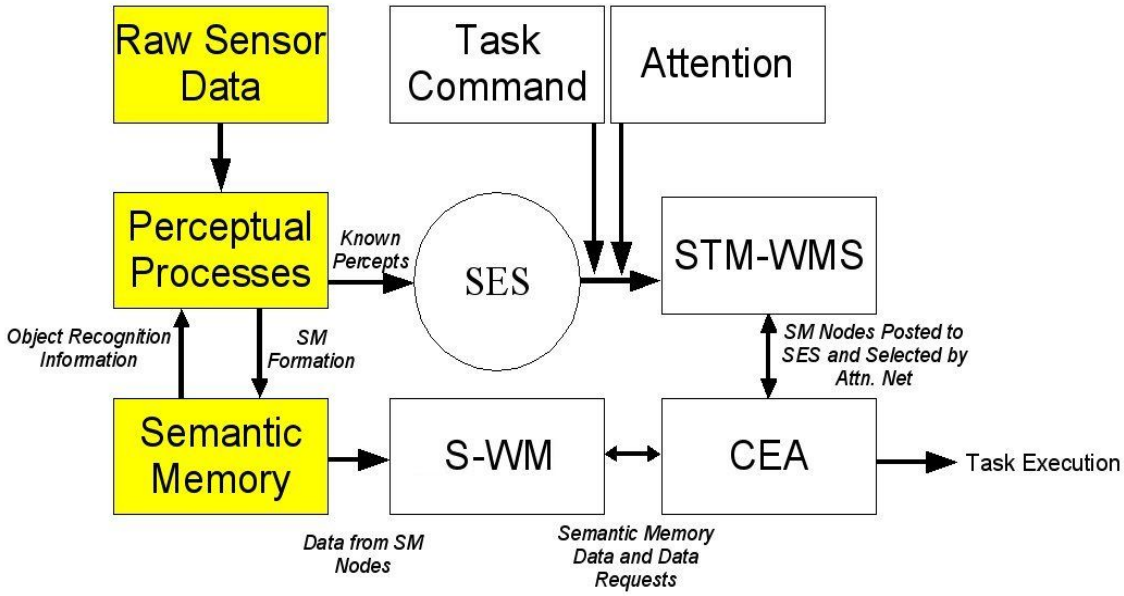

Figure 26. Semantic Memory Information Flow for Experiment

The S-WM is retrieved through an id-based query. This process is similar to the one that is used by the perceptual process to retrieve memory units in that the database is essentially accessed directly, so the process for populating the S-WM is demonstrated through this experiment. Several sample queries will be generated to show the functionality of the S-WM with respect to ISAC's executive processes as detailed in Chapter VII.

The SM consolidation technique of pruning SM units without Object Recognition nodes will not be examined since it is able to be accomplished by a single database query.

Semantic Memory Experiment 2

The second Semantic Memory experiment is intended to detail the process involved with novelty detection. This analysis will detail which stimuli, exactly, the novelty detection in ISAC's perceptual processes should return a positive value. This process is detailed in Figure 27.

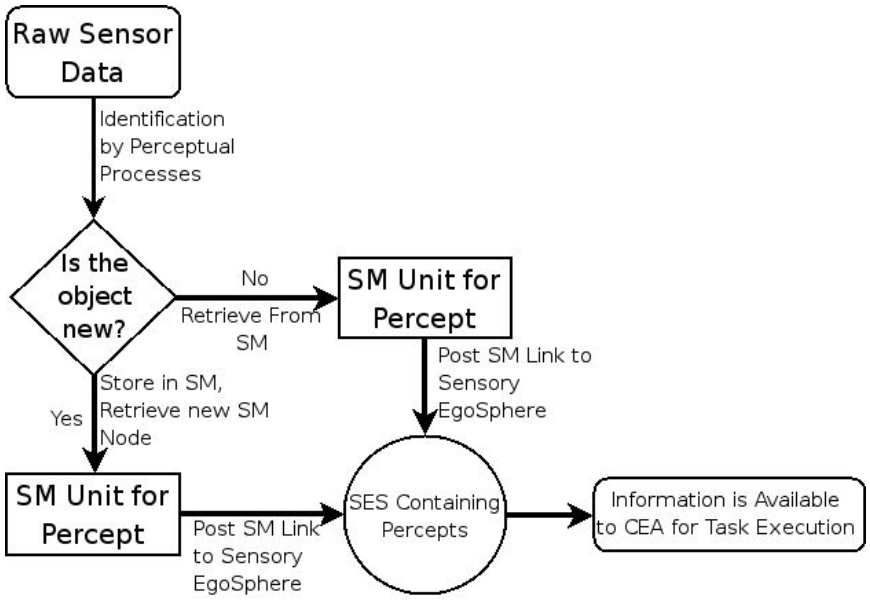

Figure 27. Novelty Detection Decision Process for SM Experiment 2

Episodic Memory Experiment

Several sample Episodic Memories will be generated, and it will be shown how the system will perform given a set of percepts. The Episodic Memory can not be tested on-line in real situations due to the lack of Central Executive and Emotion Agents,

however, it is important to demonstrate the performance of the system in a variety of circumstances for future system integration. Four experiments in particular will be discussed to highlight the contribution of several elements of the algorithm.

The decay process of the memories will be simulated as well and some acceptable values for constants in the system will be derived for use when the Episodic Memory system is put on-line.

Experiments 2-4 require a corpus of candidate episodes, full details of which are given in Appendix B.

Episodic Memory Experiment 1

The Episodic Memory approach will be contrasted with existing computer science approaches to database design and retrieval. Various aspects of the system will be simulated and discussed in detail.

In other words, the current approach for EM retrieval will be contrasted with a more traditional approach of storing all experiences into a large database and retrieving using a tree-based search with O(log N) retrieval time. In this case, as in the later descriptions containing big O notation, N represents the number of units unless otherwise specified.

This comparison will be accomplished through analysis of the EM retrieval method's space and time complexity versus that of a traditional database system operating in an environment in which memories are formed at a linear rate.

Episodic Memory Experiment 2

The second Episodic Memory Experiment will show how the EM retrieval algorithm performs when confronted with a situation in which the desired episode gives information about a stimulus that was encountered far in the past, but was remembered due to its rarity and emotional significance.

Distractions to this search include more recently formed episodes and episodes sharing more semantic memory units with the present situation.

Episodic Memory Experiment 3

Episodic Memory Experiment 3 is designed to show how the Episodic Memory system performs when the Central Executive Agent needs to know the location of a recently lost object. This shows how the Episodic Memory system is able to retrieve a memory that is important because of its temporal relation to the stimulus instead of its emotional salience or relative scarcity.

Distractions to this search include other, less recent nodes sharing more SM units in common with the current situation and nodes with lower decay values.

Episodic Memory Experiment 4

The fourth Episodic Memory experiment shows how ISAC is able to retrieve an episode based on its utility to the Central Executive Agent. For example, if the CEA requires a previous plan execution, it should be able to guide the E-WM system to retrieve an episode in which that plan was executed with less consideration given to the current environment. This shows how the Episodic Memory system is able to act based solely on executive context supplied by the CEA.

Distractions to this search include more recently formed nodes, nodes with more emotional salience, and nodes with as many or more common SM units.

CHAPTER VII

EXPERIMENTAL RESULTS AND ANALYSIS

This chapter details the results of the experiments proposed in the previous chapter for testing and analyzing the performance of three memory systems proposed for ISAC's cognitive architecture. It also contains an analysis of the results.

Procedural Memory Experiment 1

For the first experiment, the Procedural Memory (PM) was tested through a simple run of the modular controller (Chapter III) under a simulation of ISAC's hardware system. The objective of this experiment is to test how ISAC can point to a location not accessible by interpolation of any one of the behaviors stored in the PM. Thus the modular controller is forced to realize a motion through the combination of several behaviors.

The Procedural Memory system for this experiment contains five dissimilar behaviors: *reach out, reach up, swing arm, wave,* and *handshake*. ISAC was then given the task of pointing to a location that could not be reached through the interpolation of any one of the stored behaviors. Reaching to the point requires the interpolation of any of several combinations of all three behaviors. The system must find one in which the three behaviors are shared, and in which the reward for each behavior is sufficiently high.

The movement was executed within the PISAC simulator, a static simulator designed to test ISAC's movements.

Encoding and retrieval of PM units are tested in this experiment. If the system executes the requested movement from stored behaviors, it logically follows that the retrieval and combination of PM units is correctly implemented. This encoding has also been tested through comparison of data within the database and the text files that are parsed to populate the database with movement coefficients.

As discussed in Chapter III, the retrieval of PM units is accomplished through a simple reinforcement learning technique. It is important to note that this reinforcement learning technique starts in a random state and takes several iterations to stabilize to an appropriate prediction of expected reward. This causes a period of unsuccessful task executions where the reinforcement learning system adapts to the reward function and learns which behaviors provide good reward.

If the system learns to select appropriate behaviors over several executions of the task, then the behavior selection mechanism has learned to approximate the reward provided by the feedback from the modular controller. Table 7 shows how the system performed in this behavior selection over several trials.

Table 7 shows the performance of the behavior selection system over several trials. A successful trial is defined as one in which ISAC successfully pointed to the target location.

Table 7. Evolution of the Modular Controller Behavior Selector [Ratanaswasd et al., 2005a]

Trial #	1. ReachOut	2. ReachUp	3. Swing	4. Wave	5. Handshake	Result
1	0.45	0.46	0.55	0.48	0.45	Success
2	0.45	0.50	0.49	0.37	0.45	Failure
3	0.34	0.38	0.37	0.37	0.45	Failure
4	0.34	0.28	0.37	0.28	0.34	Success
5	0.29	0.28	0.41	0.28	0.34	Success
6	0.25	0.28	0.43	0.28	0.33	Failure
7	0.25	0.21	0.33	0.28	0.25	Failure
8	0.19	0.21	0.24	0.21	0.25	Failure
9	0.19	0.16	0.18	0.21	0.19	Failure
10	0.14	0.16	0.18	0.16	0.14	Failure
11	0.14	0.12	0.14	0.12	0.14	Success
12	0.14	0.12	0.23	0.12	0.19	Success
13	0.15	0.12	0.31	0.12	0.22	Success
14	0.15	0.12	0.36	0.12	0.25	Success
15	0.15	0.12	0.40	0.12	0.27	Success
16	0.15	0.12	0.43	0.12	0.28	Success
17	0.15	0.12	0.45	0.12	0.30	Success
18	0.15	0.12	0.47	0.12	0.30	Success
19	0.15	0.12	0.48	0.12	0.31	Success
20	0.15	0.12	0.49	0.12	0.31	Success

Table 7 shows how the system evolved over the number of trials executed. The numbers listed beside the trials indicate the predicted relevancy of each behavior. The behaviors in red (or light gray if viewed in monochrome) are the ones selected for inclusion in the P-WM.

At first, before the reinforcement learning system has adapted to correctly estimate the reward function, the system could not consistently execute the pointing task. After the tenth trial, however, the system learned how to successfully point to the assigned location, and the reward predictor settled into a steady state.

Table 7 also shows that the reward system functions correctly. As discussed in Chapter III, the reward is calculated by the average usefulness of the behavior during the execution of a task. Figure 28 shows how different behaviors are combined during the execution of the task, a process which leads to the calculation of the reward.

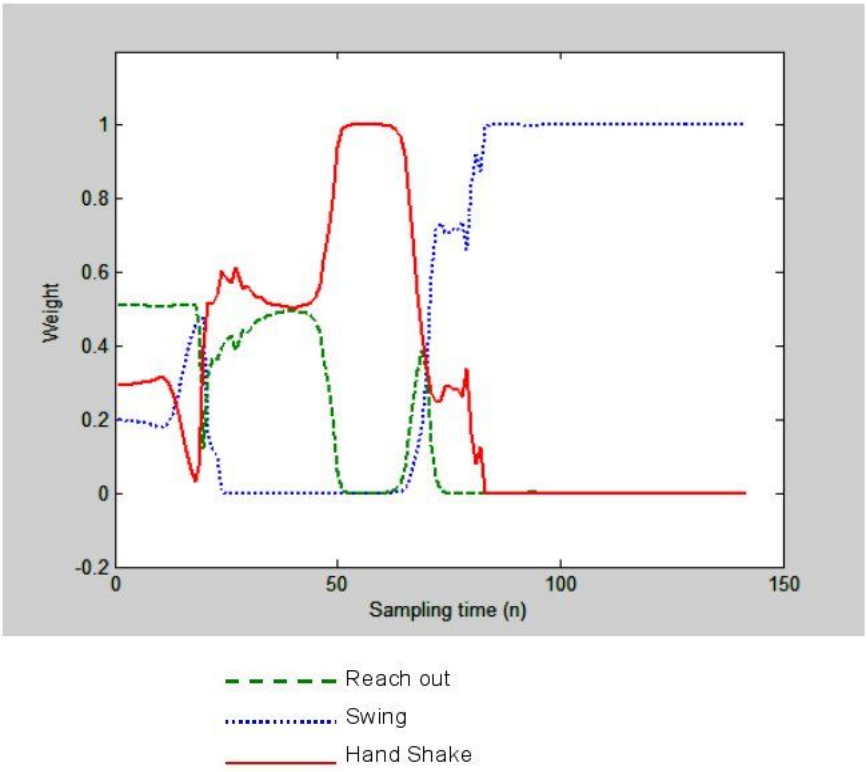

Figure 28. Change in Behavior Weights During Trial Execution [Ratanaswasd et al., 2005a]

Figure 28 shows how the weight, or relevancy, of each behavior changed over the course of a single successful pointing task execution. For example, the *Swing* behavior was used almost completely at the end of the reaching task, meaning that the target was on a path moving from a combination of the first two motions to the end of the swing motion. All three behaviors are used and combined at different times during the execution of a task. This indicates that each behavior has a certain section in which it is the best for the pointing movement.

Procedural Memory Experiment 2

The second Procedural Memory experiment is intended to show the effect of the addition of more behaviors to the retrieval system for the P-WM. The results of this experiment are detailed in Table 8.

Table 8. Results of PM Experiment with 7 Candidate Behaviors

Trial #	1. ReachOut	2. ReachUp	3. Swing	4. Wave	5. Handshake	6. Shrug	7. Skiing	Result
1	0.51	0.48	0.46	0.45	0.43	0.52	0.56	Failure
2	0.38	0.48	0.46	0.45	0.43	0.39	0.42	Success
3	0.38	0.52	0.42	0.35	0.43	0.39	0.42	Failure
4	0.38	0.39	0.32	0.35	0.32	0.39	0.42	Failure
5	0.38	0.29	0.32	0.35	0.32	0.29	0.32	Failure
6	0.29	0.29	0.32	0.26	0.24	0.29	0.32	Failure
7	0.29	0.29	0.24	0.26	0.24	0.22	0.24	Success
8	0.22	0.22	0.24	0.20	0.24	0.22	0.24	Failure
9	0.22	0.22	0.36	0.20	0.25	0.22	0.18	Success
10	0.22	0.22	0.27	0.20	0.19	0.16	0.18	Success
11	0.21	0.31	0.26	0.20	0.19	0.16	0.18	Failure
12	0.20	0.38	0.25	0.20	0.19	0.16	0.18	Failure
13	0.20	0.29	0.19	0.15	0.19	0.16	0.18	Failure
14	0.15	0.22	0.14	0.15	0.19	0.16	0.18	Failure
15	0.15	0.16	0.14	0.15	0.14	0.16	0.13	Success
16	0.15	0.12	0.14	0.11	0.14	0.12	0.13	Success
17	0.15	0.12	0.24	0.11	0.19	0.12	0.13	Success
18	0.15	0.12	0.31	0.11	0.22	0.12	0.13	Success
19	0.15	0.12	0.36	0.11	0.25	0.12	0.13	Success
20	0.15	0.12	0.40	0.11	0.27	0.12	0.13	Success
21	0.15	0.12	0.43	0.11	0.28	0.12	0.13	Success
22	0.15	0.12	0.45	0.11	0.30	0.12	0.13	Success
23	0.15	0.12	0.47	0.11	0.30	0.12	0.13	Success
24	0.15	0.12	0.48	0.11	0.31	0.12	0.13	Success
25	0.15	0.12	0.49	0.11	0.31	0.12	0.13	Success

For the second PM experiment, two new behaviors were added: moving the arm in the motion one would make when skiing with a pole, and swinging the shoulder laterally. Both movements were cyclic and at a vertical level well beneath the target point. The addition of these two behaviors to the five with which the first experiment was executed caused the settling time for the system to jump from approximately 10 trials to

15 – with the addition of two behaviors, five trials were added to the settling time of the system.

In other words, the addition of more behaviors causes the performance of the behavior selection system to degrade as the system must learn which behaviors to select based on trial and error.

It is important to note that in the results from both of these experiments there were instances where the same behaviors were selected for inclusion in the P-WM when the result was different (success and fail). These result from small irregularities in the arm kinematics, and indicate that the result of the reaching motion was on the edge of the region considered to be a movement success. Although undesirable, the effect that this has is of lowering the reward of behavior sets that are on the border of successful task completion.

This result indicates the need for a behavior selection system that is able to generalize between movement tasks to prevent the system from starting randomly for each movement task. One solution to this problem is the implementation of the Working Memory Toolkit [Skubic et al., 2004], which allows for the encoding of current state in the memory selection process. A difficulty encountered in this implementation is the encoding of both memory chunks and states into (preferably) binary vectors for use with the neural networks. This will be discussed more in Chapter VIII.

Semantic Memory Experiment 1

Because the Semantic Memory system design is simple, the analysis of the SM consists of a demonstration of an operational perceptual system using the SM to identify objects. The first experiment with the SM consists of a sample perceptual module that

attempts to detect brightly colored objects in ISAC's workspace and identify them based on information stored in the SM database.

Figure 29. Image of ISAC's Workspace Taken From Head Camera

The first step in the perceptual process is the acquisition of an image from ISAC's camera head. A sample image is seen in Figure 29. This perceptual algorithm is monocular, however the perceptual algorithm could be executed with images from both cameras at any given time to enhance its accuracy by consolidating parameters of multiple object identifications. This combination of camera data would also be useful for estimating the depth of an object for executing tasks interacting with that object. After the image was captured, it was saved to a file and the perceptual program was called.

The first filter applied to the image looks for pixels with saturations or values over a certain threshold in the HSV color space. The saturation of a pixel represents its amount

of color while the value represents that pixel's intensity. Therefore, this first filter segments out objects that are both bright and colorful.

The second filter removes pixels that are below a certain brightness threshold. Both of the thresholds were chosen by trial and error using sample images of ISAC's environment to be values maximizing the amount of pixels in objects that pass through the filter, while minimizing the amount of noise that is passed.

The third filter attempts to eliminate noise by removing all objects smaller than a size threshold. It accomplishes this through the use of a recursive “tainting” function that counts adjacent pixels as belonging to an object if they are within two pixels of a pixel already belonging to that object.

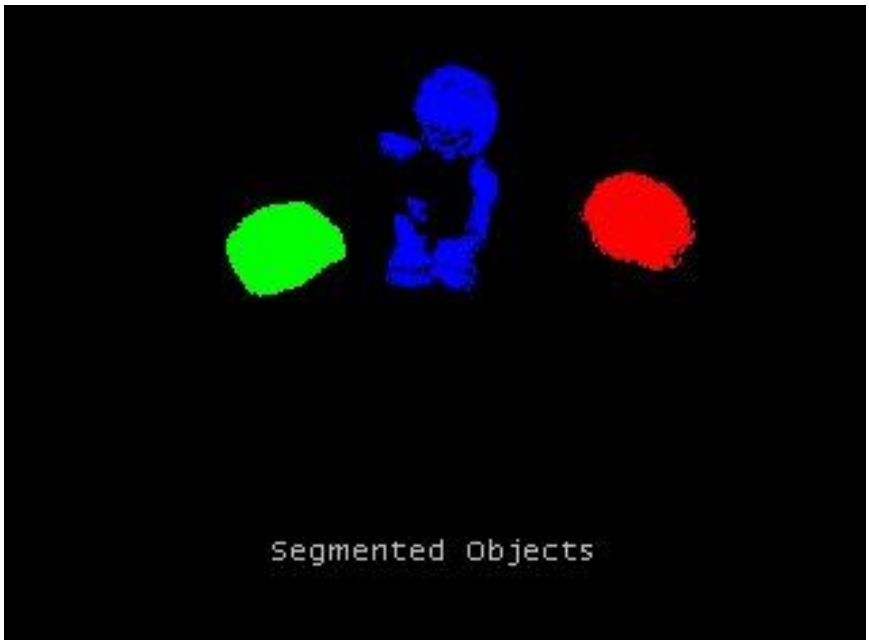

Figure 30. Segmented Objects

Figure 30 shows the three objects found in the picture in Figure 29. The object images are used as a mask over the original image to derive statistics about the object image data.

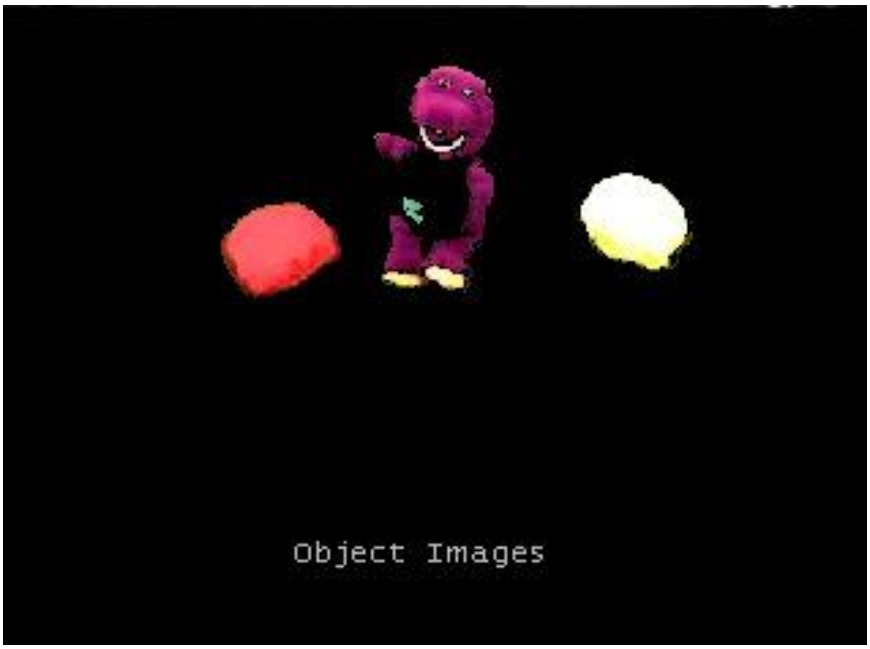

Figure 31. Object Color Contents

Figure 31 shows the color contents of the three objects that were identified in this experiment. These are the data that are used to find statistical coefficients that will be used to identify objects as discussed in SM Experiment 2.

These three objects are relatively distinct from each other, and obviously contain very different values representing these colors. As such, color-based recognition should produce effective object recognition for this set of objects.

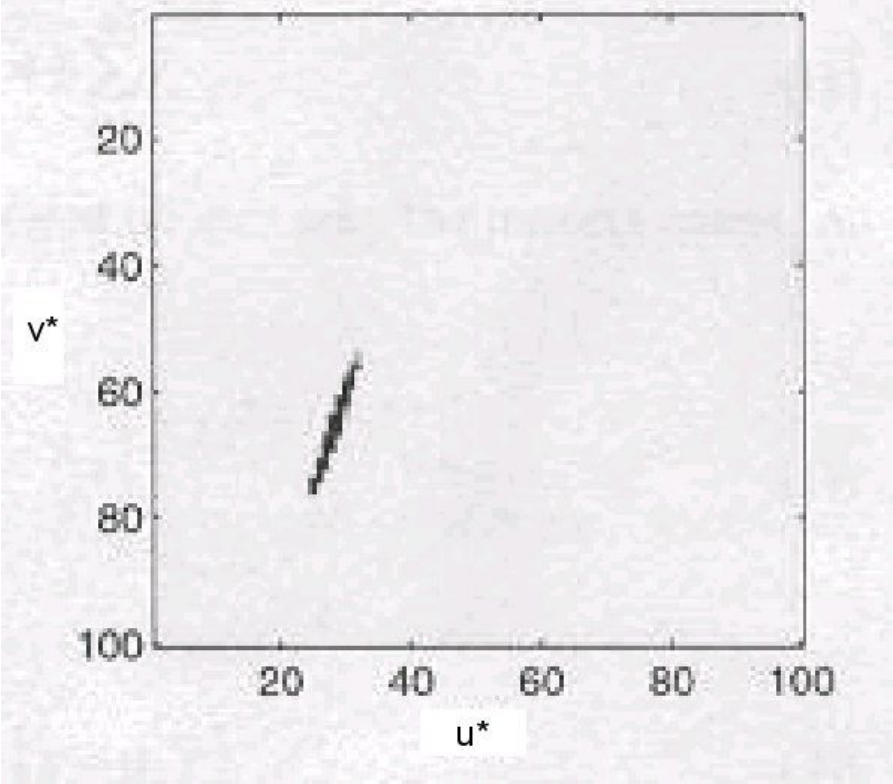

Figure 32. Sample Histogram of LUV Color Values [Barille, 1997]

The L*U*V* color space contains two chromaticity values (U and V) and a luminance value (L). These values are dimensionless, but represent a color when combined. The space was designed to model the human color visual system independently of the lighting conditions in the environment. Figure 32 shows a sample histogram of an object in the U*V* color space [Barille, 1997]. The cluster formed by the histogram is statistically modeled and stored in the RecNode SM structures in this perceptual system.

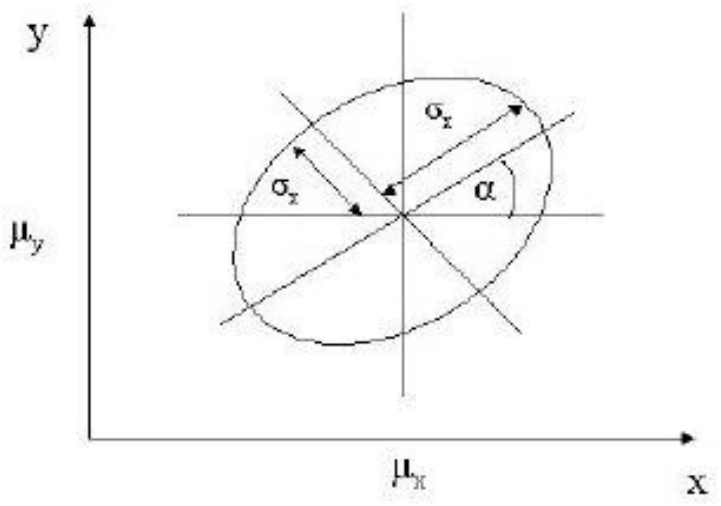

Figure 33. Color Ellipse Parameter Diagram [Barille, 1997]

After the objects were identified, they were processed to identify their U*V* color ellipses in the L*U*V* space as defined in [Barille, 1997]. Figure 33 shows an example ellipse. The U*V* color ellipse parameters are the data that was stored in the RecNode structure (See Chapter IV for more information on SM structure). As seen in Figure 32, the color information from relatively homogeneously colored objects is closely packed in a region of U*V* space. This makes the color ovuloid an effective representation of an object's coloration for recognition.

Finally, a third program takes the LUV values from the object recognition program for all objects in the picture and compares them with existing U*V* oval recognition nodes to find matches. Parameters are compared using Cartesian distance. If the oval parameters are closer than a stated threshold, the values are averaged in with the existing node (the existing data is weighted with the number of recognitions in that access, so the color weights are averaged through all exposures over time to generate a generic U*V* oval). This process is covered in greater detail in the second experiment.

With preexisting knowledge of the above three objects, the system correctly identified all three objects and averaged their oval parameters together. This identification shows that the system could generalize between images to identify objects based on color. It also shows that the system was able to pull interesting objects from a visual image of ISAC's environment for identification by the more detailed color analysis algorithm.

Table 9. Results from SM Experimental Trials

Object Name	***Trial 1***	***Trial 2***	***Distance***
Barney	53.4795, 27.5378, 24.9307, 22.5021, 2.62362	53.3927, 25.8128, 22.3735, 23.1116, 2.66371	3.0648
Red Bag	76.2444, 14.0745, 16.4845, 6.13151, -10.6122	67.5693, 13.4827, 15.6148, 6.61452, -8.81723	8.9007
Yellow Bag	1.30762, 15.617, 1.92123, 18.9472, -0.536968	0.996797, 12.5957, 1.71894, 18.8267, -0.361276	3.1559

Table 9 shows the results of the object recognition experiment. Trial 1 is the color ovuloid information stored for the initial recognition of the objects. The first two values represent the centroid of the ovuloid, the next two represent the length of the axes, and the last represents the tilt angle of the ovuloid in radians.

This information is that which was stored for each of the objects in the initial SM nodes that were created for each object. Trial 2 shows the color ovuloid parameters calculated for each object in another image when the objects were in different positions.

When the Euclidean difference is taken from the first image to the second, the values in the Distance column are generated. It can be seen that these values are very close for a 5-dimensional space, where the values for distance range well over a hundred and those for angle are in the range of -pi to pi.

The retrieval processes involved in populating the S-WM were also evaluated through a demonstration of function. The node representing Barney was requested from the S-WM by a simulated CEA, which could have received a pointer to the SM unit through the E-WM or ST-WM. The S-WM system then returned the contents of the correct node. In summary, the Barney SM node was created by an independent perceptual process and retrieved through the S-WM system to the simulated CEA.

The detailed perceptual system will obviously not operate in situations for which it was not designed, for example the identification of an object under a colored light. The system also will not recognize an object that is outside of its threshold as described in the next experiment.

This reliance on color does not provide the sort of perception that is desired in a humanoid robotic system. There are many objects that are multi-colored, and many others that are colored similarly to mundane objects such as desks and chairs in ISAC's environment. For the time being, however, ISAC can make do with color information. There are enough brightly colored objects in ISAC's environment to create goals and interaction tasks without having to recognize objects through other modalities.

Semantic Memory Experiment 2

The second SM experiment shows how the system evaluates novelty in the environment. The generic process used to achieve such detection is detailed in Figure 27 and mentioned in the discussion of the first experiment.

The first step a perceptual agent must perform when detecting novel objects in ISAC's environment is that of attempting to identify objects in the environment that fit the description of objects that the perceptual system was created to identify. In the specific example shown in SM Experiment 1, the system identifies bright, saturated objects from an image. Figure 34 shows the output of the perceptual system for a camera input of ISAC's table.

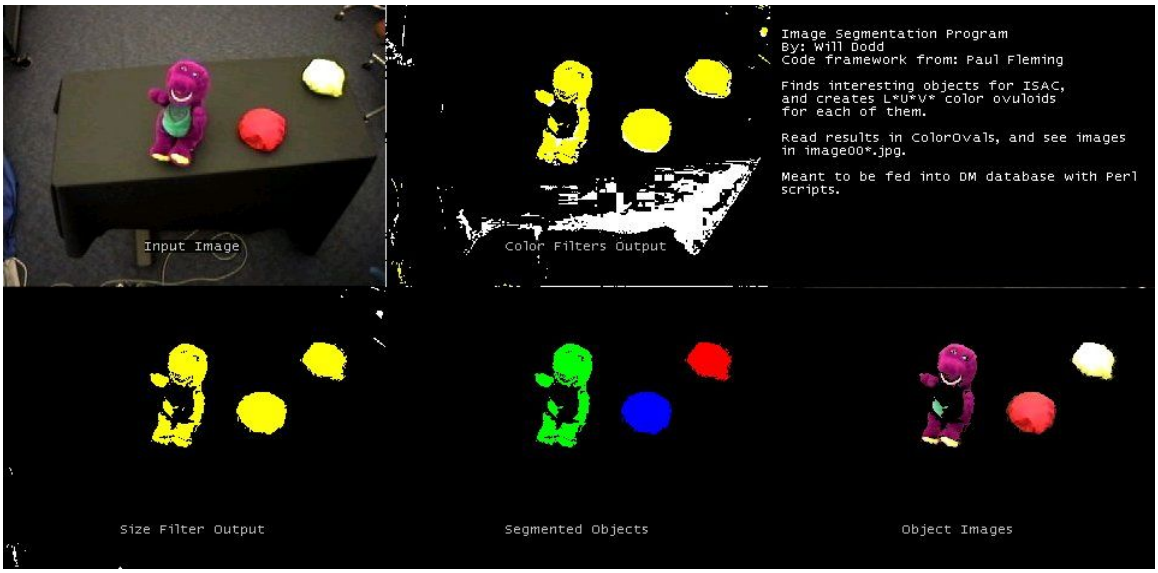

Figure 34. Parsed Image from ISAC's Camera for Initial Set of Objects

Figure 34 shows the parsed image from ISAC's environment showing three objects to be identified by the perceptual system. The frame in the top-middle shows the output of the color filter. The bottom left frame is the output of the color filter passed

through the size filter. Finally, the frame in the bottom middle of the figure shows the output of the object-segmenting algorithm.

Assuming the system does not have any initial knowledge of the identity of these objects, each of the three objects would be identified as unknown to the system. A human would then queried for the identity of each of the objects, or each object would be left nameless, to be recognized in the future as the same memory structure, but without context such as the object's name supplied by a human.

In this experiment, none of the three objects was known to the perceptual algorithm, so each object was identified as unknown to the system. The human was then queried for the identity of each found object using the information in the Exemplar Node for the newly created object (Chapter IV details the function of the Exemplar Node). Figure 35 shows the perceptual program querying a human operator for the identity of the unknown Barney doll shown in the scene in Figure 34.

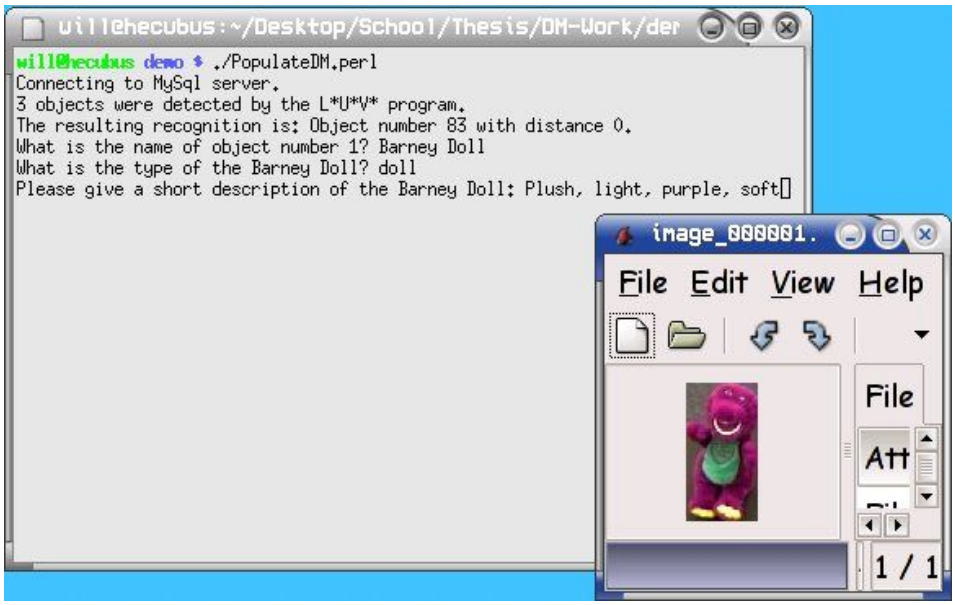

Figure 35. Unrecognized Object Node Creation

In the detailed perceptual system, if the color oval parameters are not similar to those in any existing node then a new Object node is created and populated with an image of the object placed into the ExampleData node. The oval parameters are then placed into a RecInfo node.

Figure 35 shows the SM node creation user query. This process only occurs when the recognized object does not fit any object previously in the database. In other words, it occurs when novelty is detected. In future design of perceptual processes, the process could instead create a "stub" node in the STM so cognitive processes could query the user to identify an unknown object.

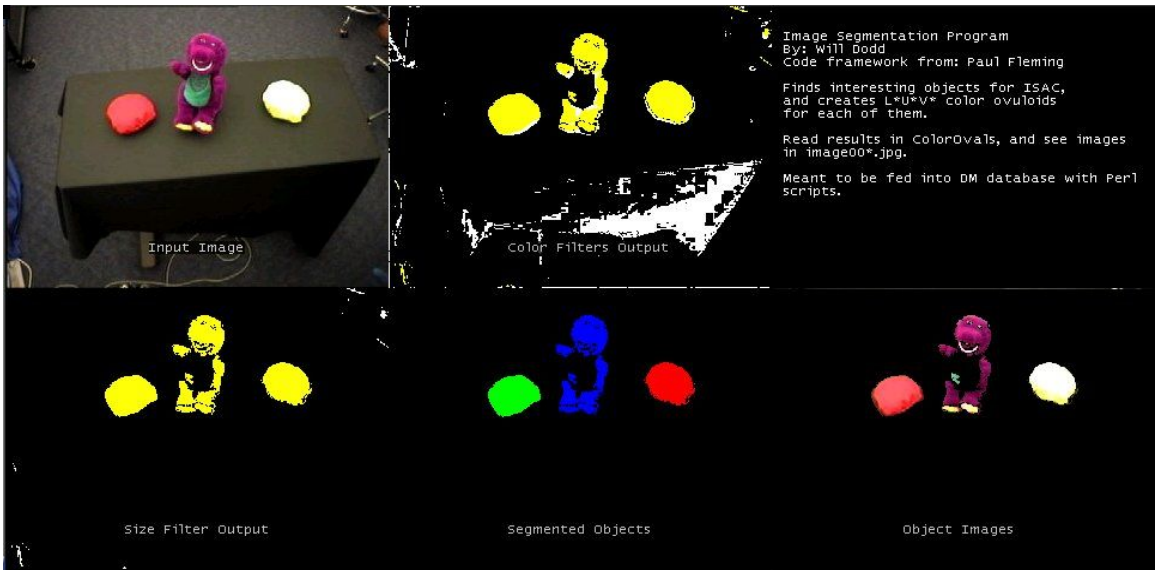

Figure 36. Parsed Image from ISAC's Camera for Second Set of Objects

Figure 36 shows a scene similar to that in Figure 34 with similar objects placed in different locations. The novelty system should detect that each of these objects has been seen before and recognize each one as being associated with a particular SM node. This

node would then be posted to the Sensory EgoSphere. If one of the objects had not been seen before, the process for creating a SM node would be executed once more as discussed about Figure 35.

In the experiment, the system was able to correctly identify all three objects, and associate them with their corresponding SM units for posting to the SES.

The specific method through which this experiment accomplished this categorization task was through comparison of the Color Ovuloid parameters. If the Cartesian distance between the parameters of the object to be identified and those of any other SM node in the database is above a threshold, then the detected object is said to be dissimilar enough to existing nodes to warrant the creation of a new object node.

The setting of such a threshold should be carefully considered in production, as it defines the regions of similarity and tolerance for sensor error for each object in ISAC's environment. It is probably best to set the threshold to the smallest value that gives accurate results, because ISAC is located in a sealed laboratory with fixed lighting to preserve the object recognition values.

Table 9 details typical distance values one would see for the detailed perceptual system in normal operation with ISAC. A threshold value of 15 was set for the experiment, although the maximum deviation for like objects over the course of several image parses was around 10, the distance between the parameters of unlike objects was above 50. 15 is therefore a reasonable value for setting the cutoff, erring on the side of safety in avoiding misclassification, which could skew the recognition parameters of a particular SM node.

In the perceptual process detailed above, found objects have their parameters averaged together to produce a more generalized object. The tolerance for such averaged

objects is then increased in order to provide a greater chance of categorization of common objects under non-standard lighting conditions or with sensor error.

The exact process for this perceptual classification problem, however, is highly domain-specific and should be determined for each modality. For example, the sort of deviation due to environmental conditions that one would find when identifying objects by color is different than that found when attempting to identify objects by shape.

Another caveat to this approach is that of perceptual algorithms that can identify the same object with more than one recognition node. For example, the Barney could be identified by either the color of its deep purple coat or its sea-foam green stomach. It would not make sense to average these two modalities together, nor would it make sense to use a simple threshold to identify the Barney if both indicators were required. Great care must be taken to account for these types of circumstances, and perceptual algorithms should be designed to match ISAC's environment – just as human perceptual processes evolved to fit our environment.

One way to turn this weakness into a strength is to provide a communications interface between the perceptual agents so that they may request identification of objects by other agents that own an Object Recognition node on the same object. In addition to allowing for more robust object recognition through collaboratively identifying the same object, this technique could allow for averaging of the percepts generated by the perceptual agents, allowing the system to identify similar objects through an aggregate of results.

Once ISAC's full system is operational, the experience of learning the identity of objects will be contained within an Episodic Memory unit. Each of the newly learned Semantic Memory units will provide a new atomic unit for the composition of Episodic Memory units.

Episodic Memory Experiment 1

The first analysis to be detailed about the Episodic Working Memory is the setting of the history constants v and b, and the derivation of the decay constant alpha from emotional salience. These constants affect the retrieval performance of the system because they are linked to both the likelihood of retrieval of a particular memory and the total size of the memory database.

As stated in Chapter V, the history component is defined as:

$$P(A|H_A) = \frac{v+n}{M(t)+b} * r(t) \qquad \text{(5.3, Chapter V)}$$

In Equation 5.3, v and b are constants, n is the number of accesses, and the decay equation, r(t), is defined by an exponential decay containing the alpha decay constant set by the emotional salience of the memory. The decay formula r(t) is:

$$r(t) = \alpha e^{-\alpha * t} \qquad \text{(5.4, Chapter V)}$$

The history component of Episodic Memory for a variety of decay constants may be seen in Figure 23 in Chapter V. This figure shows how a single Episodic Memory unit

would decay with different alpha decay constants representing different emotional saliences and no accesses during the decay process. Memories with low decay constants decay at a much slower rate than those with high constants, with marked differences at higher time values. Episodic Memories with high emotional content, therefore, should have low Alpha values. To accomplish this relationship, alpha is set to be the salience subtracted from one where the salience is a value within the range of zero to one.

Setting the decay to equal this formula will allow for intuitive generation of the salience value because a higher salience value indicates higher importance, and simple debugging of the link between Episodic Memory and Emotion because there is a linear relationship between a memory's salience and its decay constant.

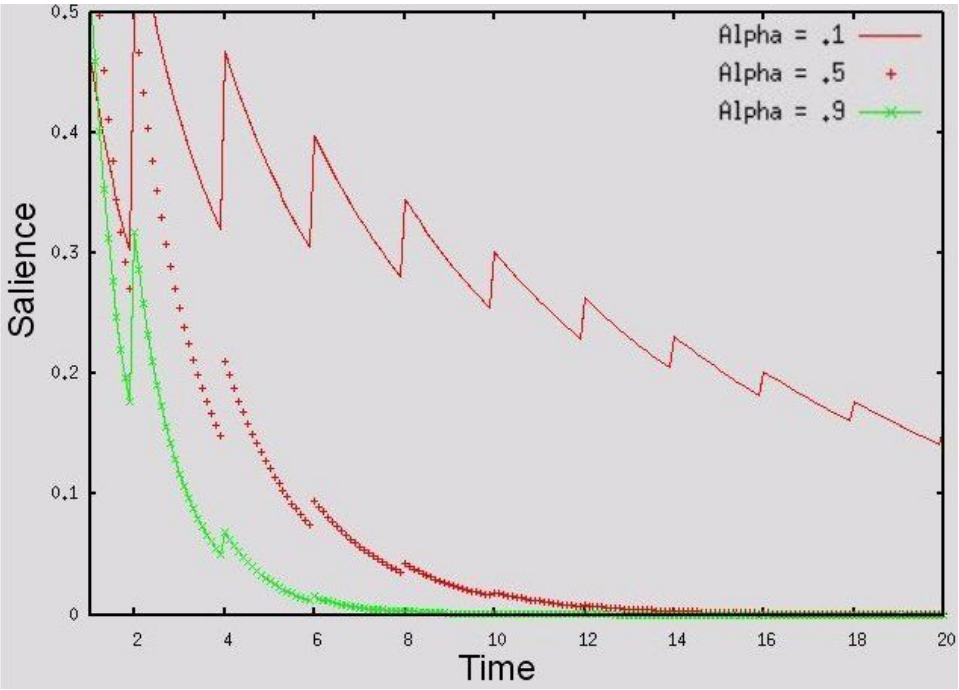

Figure 37. Episodic Memory Decay with Memory Access

Figure 37 shows the same decay process with one access to the memory every 2 time units. It may be seen that access of a memory significantly prolongs its historical relevance, especially when the memory is stored with a high salience value. This means that important memories that are accessed frequently will last almost indefinitely, especially so if many tasks are performed per time step, raising the probability that a given memory will be selected.

The next task in honing the decay formula is setting the constants v and b. These constants dictate the shape of the curve, and should be set in order to tune the system to the desired performance.

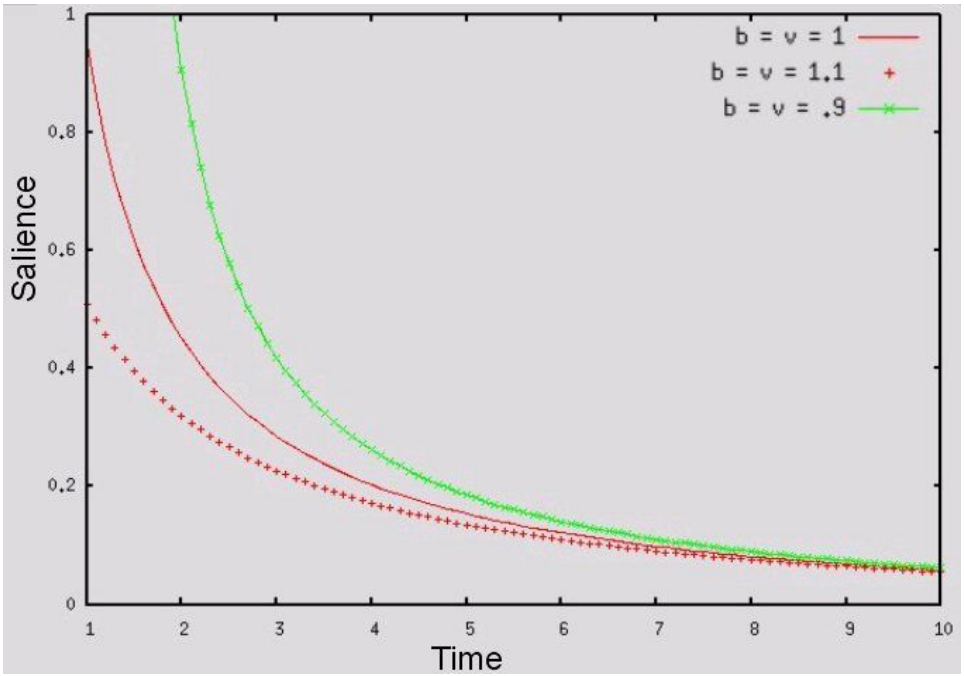

Figure 38. The Effects on EM Decay of Altering b and v

As may be seen in Figure 38, b and v effect the curvature of the decay function. Alpha and n are constant for this figure (alpha = .1 and n = 0). Heightening b and v causes

the curve to flatten out, while lowering them causes a greater drop in memory relevance due to history. For simplicity, v and b will be set at 1 for the following demonstrations. In the future, however, these values may be tweaked to cause memory decay to more appropriately match the environment of the robot.

After the coefficients of the decay process are set, the performance of the system may be measured in order to compare the system with other, more traditional database systems. The first performance metric to be measured is the growth of the size of the Episodic Memory system through time. As memories are formed in a linear fashion, the size of a traditional database increases linearly. The size of the Episodic Memory database, however, will increase at a smaller rate due to memory decay.

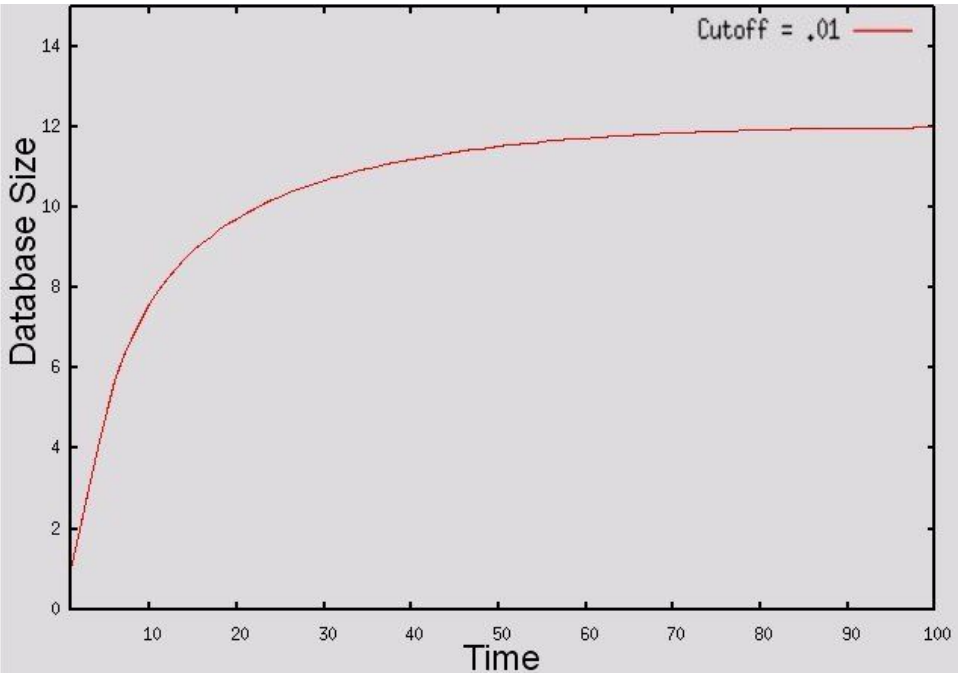

Figure 39. Episodic Memory Size vs. Time

The crucial value that influences the size of the Episodic Memory database is that of the memory cutoff. Any memory with a history that falls below this value is removed from the system. Figure 39 shows how the memory size increases with one memory formation per unit time. Figure 41 shows how different cutoff values affect this change in memory size.

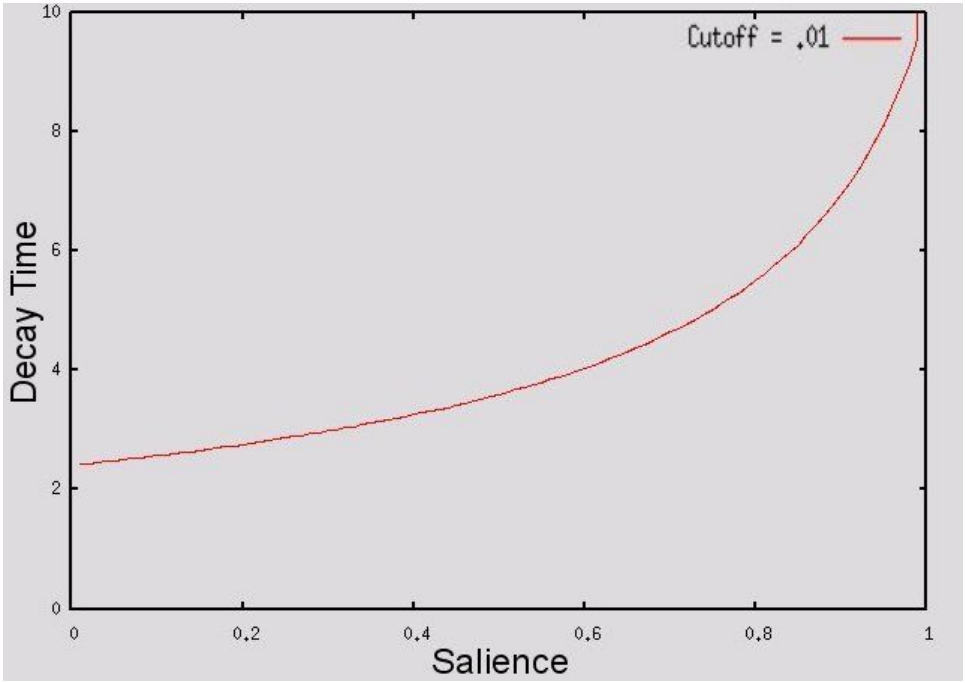

Figure 40. EM Decay with Varying Salience

Figure 40 details the decay time generated by several memory salience values. It is important to note that the rising salience of a memory prevents it from decaying for many time units. A uniform distribution of saliences between 0 and 1 was assumed for the following EM database size experiments, however any statistical distribution centered at . 5 would produce similar results because of statistical averaging.

Figure 41 shows the growth of the size of the Episodic Memory system with linear memory formation (one memory is formed per time step). At each time step, the likelihood of the decay of a memory formed at each time n is calculated, and that probability is summed to find the average value of the memory at any given time. The decay was calculated for a uniform distribution of salience values – for each time step for each memory formed at an earlier time the salience value causing that memory to decay was found. The probability that the salience of that memory is higher than the calculated value was then calculated. This probability is the probability that the memory in question is still in the database. For different distributions of the salience, the resulting asymptote is similar but the initial growth curve differs.

As may be seen in Figure 41, the memory size approaches an asymptote, and stabilizes due to the decay rate offsetting the memory formation rate. The total growth of the memory is logarithmic.

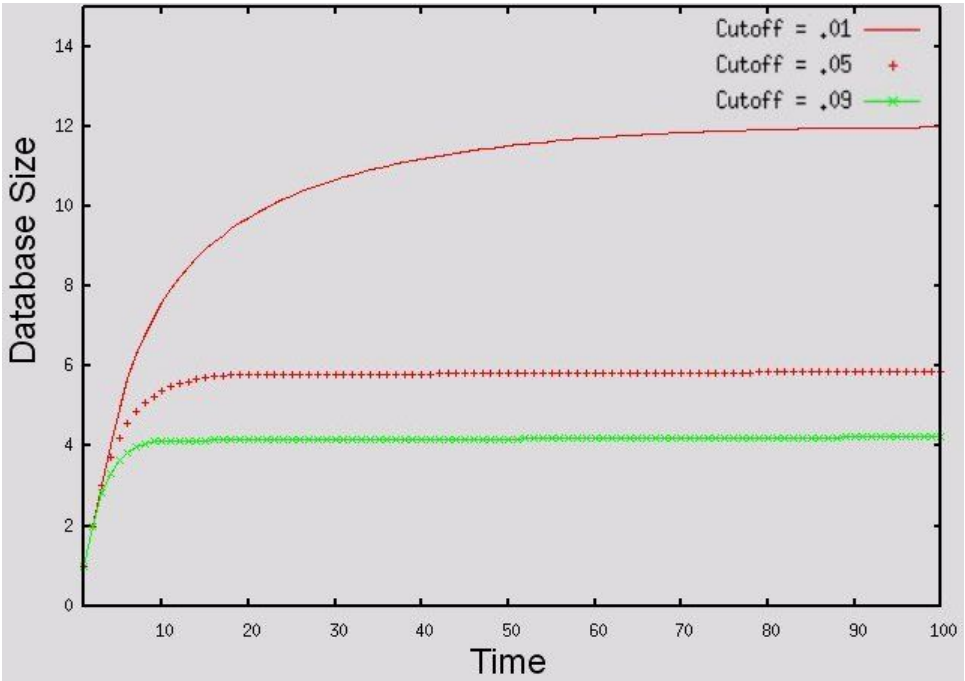

Figure 41. Memory Size Increase with Different Cutoff Values

Figure 41 shows the increase in size of the memory for several different cutoff values. The cutoff value is inversely related to the final size of the database. It is desirable, therefore, to set the cutoff value to be as high as possible without discarding a significant number of useful memories. This may be accomplished by measuring the history components of each retrieved memory during the normal operation of the system, and tracking the lowest number. With significant experience, this number may be determined empirically. It is important to note that this value should be calculated and set after significant experience is acquired – otherwise the cutoff would rise continuously and prune all episodes.

Finally, the performance of the system may be estimated and compared to the performance of a more traditional database system that stores all information and uses a powerful O(log n) search technique to find the desired values. Because the size

complexity of the database does not increase linearly with n, but remains at O(log n), the EM search algorithm is comparable in complexity to the more traditional technique with linear memory search time. This time is probably better than linear, however.

An analysis of the EM retrieval algorithm shows that, in the worst case, for each cue x, n episodes may be considered. This produces a search complexity of O(x*n), where x is the number of SM units and n is the number of EM units. The SM size is necessarily linearly related to the EM size (because SM units are pruned when they are not connected with any EM units), therefore the complexity becomes $O(n^2)$, where the size of n is O(log n). In reality, therefore, the worst-case complexity of this search then becomes O(n), larger than the O(log n) of a more traditional method of storing all information.

This analysis has some fundamental flaws, however. The two terms that affect the complexity of this algorithm are the average number of SM units contained within the memory cue, this number will be called m in this analysis, and the average number of EM units containing a given SM unit, which will be called p. The worst case complexity in this scenario is constant for the system with O(m*p). p, however is very likely proportional to the number of EM units in the database, n, which should produce an algorithm that operates in O(n). This would produce results similar to those with a more traditional approach, with the benefit of allowing the system to return results that exhibit desirable characteristics as seen in EM experiments 2-4.

Because episodes tend to contain a small number of important SM units that stay with the cognitive system for the duration of a goal completion, m and p rely on the environment of the system. If ISAC is situated within a rich environment with many SM units, the performance of the Episodic Memory retrieval system will increase due to a

smaller p because there are more SM units that have a chance of inclusion in any EM unit, so the probability that a given SM unit is contained within any EM unit is smaller.

It is important to note that the memory formation time in this technique is O(m) because back-references must be added to all SM units that constitute the EM unit being stored. The memory formation time in the tree-based search method is O(log n). Although episode formation does not occur with great frequency, this does represent a small performance gain over traditional tree-based search techniques.

The second part of the analysis of the Episodic Memory system is to show examples in which the capabilities of the Episodic Memory retrieval system are highlighted. Three scenarios are proposed to demonstrate the operation of the Episodic Memory and to demonstrate the reasoning behind including each factor used for the retrieval process resulting in E-WM population.

Episodic Memory Retrieval Experiments

EM experiments 2-4 detail how the system is designed to retrieve the correct episode in a variety of circumstances that ISAC might encounter.

Table 10. Elements of EM Retrieval Tested by EM Experiments

EM Experiment #	***Rarity of SM Constituents***	***Contextual Relevance***	***Executive Attention***	***EM Salience***	***Recency of Formation***
2	X			X	
3			X		X
4		X	X		

Table 10 shows which aspects of the EM retrieval are most important for the following experiments. Each of these cues that tell the system how relevant a memory might be are described in turn with the analysis of their individual experiment.

Episodic Memory Experiment 2

The second proposed EM experiment demonstrates how the Episodic Memory system uses the history component (which conveys the emotional content of an episode) to return a highly emotionally salient episode. This experiment demonstrates how the episodic memory system selects highly salient episodes over episodes that are less salient.

The cognitive control experiment as discussed in [Kawamura et al., 2005] can be summarized as follows. ISAC performs a simple task (such as following a moving object around the room with its cameras). A person enters the room, and yells “Fire!”. ISAC must retrieve the episode associating the current situation with a highly emotionally salient past experience, and must then switch tasks to warn the experimenters to leave the room based on this past experience.

The specific question to be answered by the segment of this cognitive control experiment detailed in this section is: “How does the Episodic Memory system know to retrieve the correct, highly salient episode, and how does the salience of the appropriate episode change over time?” In other words, this experiment is intended to show how the emotional salience of a memory influences its retrieval process, and how the salience could be important to selecting the appropriate memory for a given context.

As a precondition of the experiment, ISAC's Episodic Memory banks were loaded with a number of episodes, both salient and non-salient, relevant and non-relevant (Appendix B). These Episodes approximate tasks that ISAC might actually come across,

such as locating or learning an object, playing a game with a human experimenter, learning new movement commands or tasks, or greeting humans. Thirty-two sample episodes were stored in ISAC's EM system, and the task of the EM-WM system is to select the three most relevant episodes to the current situation. The target episode was a previous encounter with the “Fire!” stimulus, in which ISAC was taught with high emotional salience that humans should leave the room.

ISAC's task was then to retrieve the target episode based on the contextual relevance supplied by the relatively rare “Fire!” stimulus augmented by the strong emotional salience associated with the target stimulus.

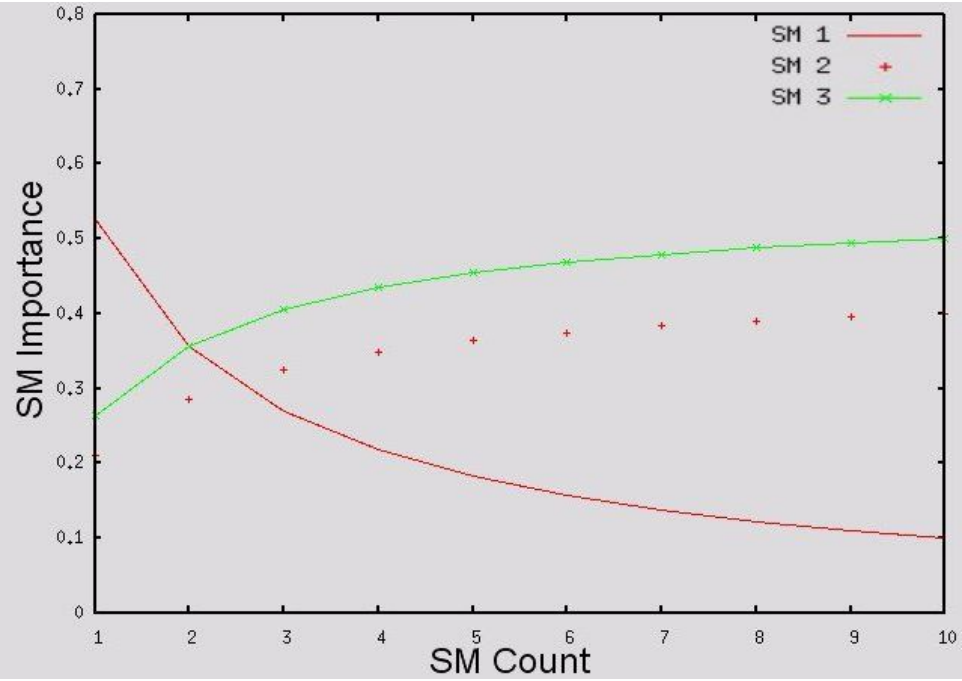

Figure 42. Effect of Number of References to SM Relevance to EM Retrieval

Figure 42 shows the influence of the count of a particular SM unit to that unit's relevance when selecting EM units. The simulation detailed in Figure 42 was run with

three SM units in the SM frequency vector of the forming EM. SM unit 1 had a frequency importance of .1, SM unit 2's importance was .4, and the importance of SM unit 3 was .5. The access count for SM units 2 and 3 was set to 10, while the access count for the first was varied from 1 to 10 to produce the graph above. When compared to the other two SM units, the relevance of SM unit 1 to memory retrieval is dramatically increased from its original value.

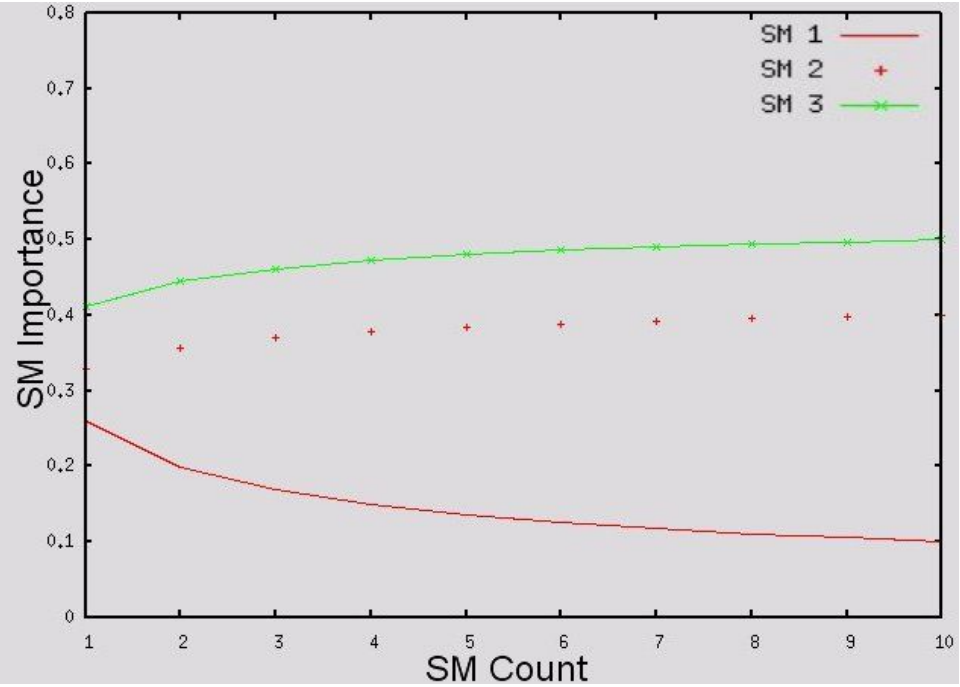

Figure 43. Effect of Number of References to SM Relevance when Square Root of Reference Count is Used

Figure 43 shows a method of reducing the influence of SM Count to the EM Fingerprint calculation. In this simulation the same contextual relevances are used as in Figure 42, however, when calculating the effect of the rarity of a SM unit to the forming fingerprint, the square root of the access count is used instead of the raw access count itself. This causes the variation between memories to drop considerably, and could be

useful in the future depending on the variance of SM count within the EM in the context of ISAC's environment and the system as a whole.

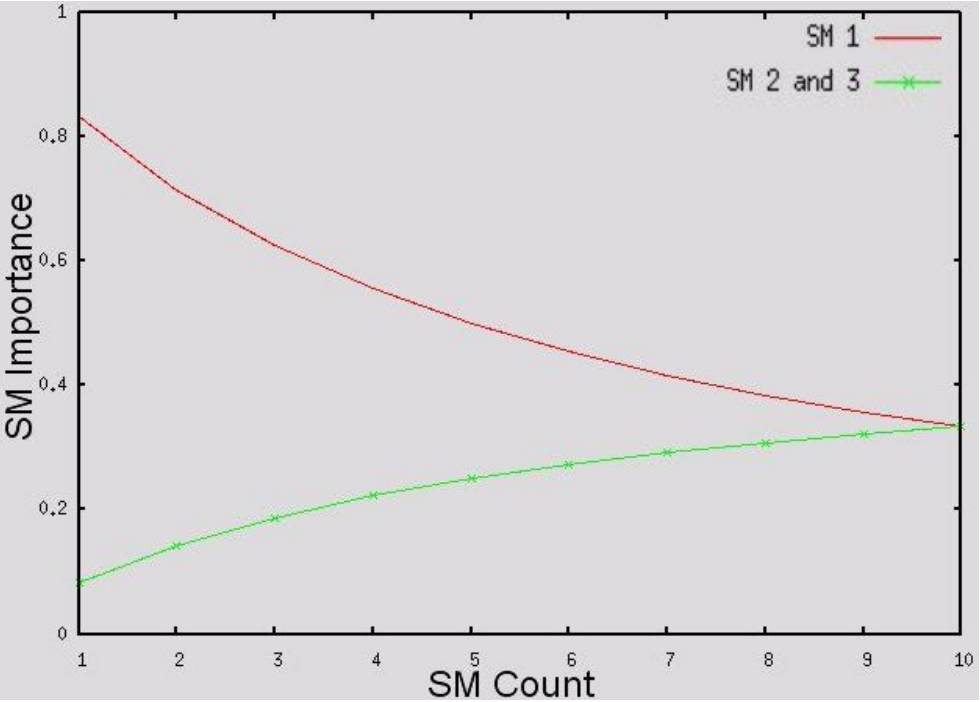

Figure 44. Effect of Rarity of SM Units of Similar Relevance

Figure 44 shows the effect of the SM rarity term on SM units with similar contextual importances. The count of SM unit 2 is held constant at ten, while the count of SM unit 1 is varied. One may see that this rarity term allows for a SM unit to hold a major advantage over the other SM units for calculation of relevance.

The cue for the memory consists of a simulated episode in which ISAC performs a task and is interrupted by a person giving the “Fire!” stimulus. The fingerprint produced by the cue shows how the system operates to retrieve the target memory. Even though the “Fire!” stimulus represented a small fraction of the forming EM unit, its rarity caused it to be weighted more than the other constituent SM units because the fingerprint weight of a

given SM unit decreases as its usage increases. The strength of the “Fire!” stimulus in the cue was then enhanced because of the strong emotion displayed during the previous exposure to “Fire!”.

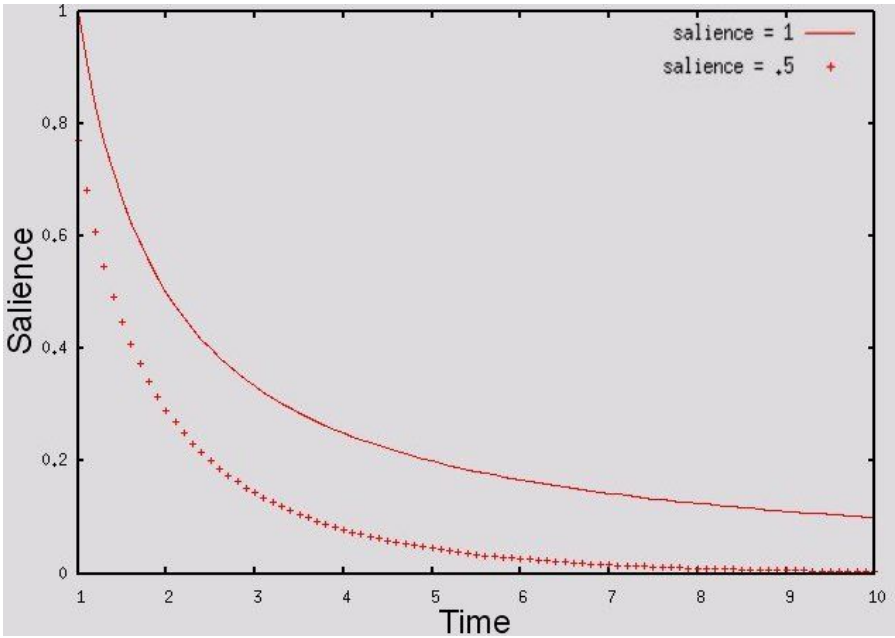

Figure 45. Comparison of Decays for EM Experiment 2

Figure 45 shows how the initial salience of the “Fire!” memory causes it to remain relevant as time progresses. The solid line shows the decay of the “Fire!” memory as detailed in Appendix B. The dotted line is the decay curve for the main group of corpus memories (with saliences of .5). This factor alone gives the “Fire!” memory an important advantage over the other memories. At the age of 3 time units, the “Fire!” memory has an approximate historical relevance of .4, while that of the other memories is near .3.

The following demonstration of the effective retrieval of the memory units was generated with the Episodic Memory corpus listed in Appendix B. The history component of the memories is set to .3 and .4 as shown in figure 45.

An example cue for the Fire! experiment can be described as:

Table 11. Cue for EM Experiment 2

Goal	***Subject***	***S-WM***	***ST-WM***
Visually Track	Red Bag	rb, visually track, Fire!	Fire!, rb, Person 1

This cue is derived from the description of the environment of the experiment as discussed earlier. In the above table, rb represents the semantic memory node for the red bag. ISAC is performing the task of visually tracking an object (in this case the red bag) and is interrupted by the "Fire!" stimulus.

This cue is translated into the following histogram by the process shown in Figure 25.

Table 12. SM Frequency List for EM Experiment 2

SM Node	***% Constituent***
Visually track	25
Red Bag	37.5
Fire!	25
Person 1	12.5

After weighting for rarity and cognitive weighting, the fingerprint shown in Table 12 is created. Since cognitive weighting does not take part in this EM experiment, the value of this term is set to one for all calculations. The rarity of each SM node was calculated through the use of the Episodic Memory table listed in Appendix B.

Table 13. Unnormalized Fingerprint for EM Experiment 2

SM Node	***Score (% importance unnormalized)***
Fire!	25 * 1 * 1 = 25
Person 1	12.5 * .33 * 1 = 14.17
Red Bag	37.5 * .25 * 1 = 9.375
Visually track	25 * .25 * 1 = 6.25

After normalization, the final fingerprint is obtained:

Table 14. Normalized Fingerprint for EM Experiment 2

SM Node	***Score (% importance)***
Fire!	45.6
Person 1	25.8
Red Bag	17.1
Visually track	11.5

The "Fire!" SM node is the most important in this fingerprint due to its relative scarcity. The weighting system that uses the full occurrence of a Semantic Memory unit instead of a root of the count for that unit was used in this experiment.

If the cognitive system were to provide attention to the "Fire!" stimulus (as might happen in a real situation in which a rare stimulus was encountered), the score for the Fire! SM node could be higher.

The following table lists the performance of the target "Fire!" episode matched against the Visual Tracking of Person 1 (VTP1) episode. Both episodes are described in Appendix B in more detail, and the technique described in Chapter V was used to calculate the score for each episode.

The VTP1 episode was selected for this demonstration because it has the next highest score from the training corpus. It is able to receive relevance from both the Visual Tracking goal and the Person 1 semantic memory unit.

Table 15. Calculation of EM Scores for EM Experiment 2

Episode	***Fingerprint***	***Occurrence***	***History***	***Score***
"Fire!"	0.456	0.200	0.400	0.036
VTP1 (P1 cue)	0.258	0.250	0.300	0.019
VTP1 (VT cue)	0.115	0.125	0.300	0.004

The final score for the "Fire!" episode was .036. The final score for the VTP1 episode was the sum of its scores, .023. Because of the rarity of the "Fire!" SM unit, the "Fire!" episode would be selected even if its history values were equal to those of the VTP1 episode.

It is also important to note that this algorithm returns the top n episodes, where n is the size of the E-WM. If the retrieval of the "Fire!" stimulus were erroneous, relevant episodes such as VTP1 or the Visually Track the Red Bag episode would still be returned for use by the cognitive system.

This scenario shows how the system was able to draw attention to information that was both rare and salient. If the target EM had been formed with a low amount of emotion, it would have decayed quickly by Equation 5.2 and would not be retrieved. The emotional information therefore allows the system to remember both recent information that was not very salient but was recently formed and information in the distant past that was extremely salient. This characteristic could be useful for both locating a recently seen object or retrieving old but important information.

In this task the correct memory contained rare elements and was emotionally salient, but was not recently formed or relevant to the current situation in terms of total similarity in constituent SM units.

The retrieval of this episode could fail under a variety of circumstances. First, if the emotion agent miscalculated the salience of the initial episode, the search would fail in almost all situations due to its decay.

Assuming correct salience generation, the memory could fail by a variety of other means. If there were other episodes with which the cue shared many common SM units, the episode that was relevant but distant in the past might be ignored for more recent episodes. Similarly, if many episodes contained the "Fire!" stimulus, the stimulus would not be as important and might be discounted extensively.

These potential failures, however, are not failures at all. The retrieval algorithm is a mathematical formula with ample opportunity for adjustment to the environment of the cognitive system. If the influence of the age of a memory should not cause the memory to decay as quickly, one needs only to adjust the index terms or salience calculation to fit the needs of the CEA. If the reference count for a SM unit is not terribly important to retrieval, adjust the fingerprint calculation function to take the square-root of the occurrence.

In short, the Episodic Memory system will retrieve only those episodes that it was tuned to retrieve.

Episodic Memory Experiment 3

The third scenario, or the "Lost Object" scenario, details how a recent EM unit is retrieved to accomplish a task. In this experiment, ISAC observed an experimenter

occluding an object, a process that was simulated and stored in Episodic Memory. ISAC was later queried for the location of the object, under the scenario that the episode must contain the information needed to glean the location of the object so ISAC could point to it. The correct episode must therefore be retrieved through contextual similarity and executive attention as well as recency of formation. The retrieved episode is not especially salient, nor does the recorded episode contain the same goal as the memory cue.

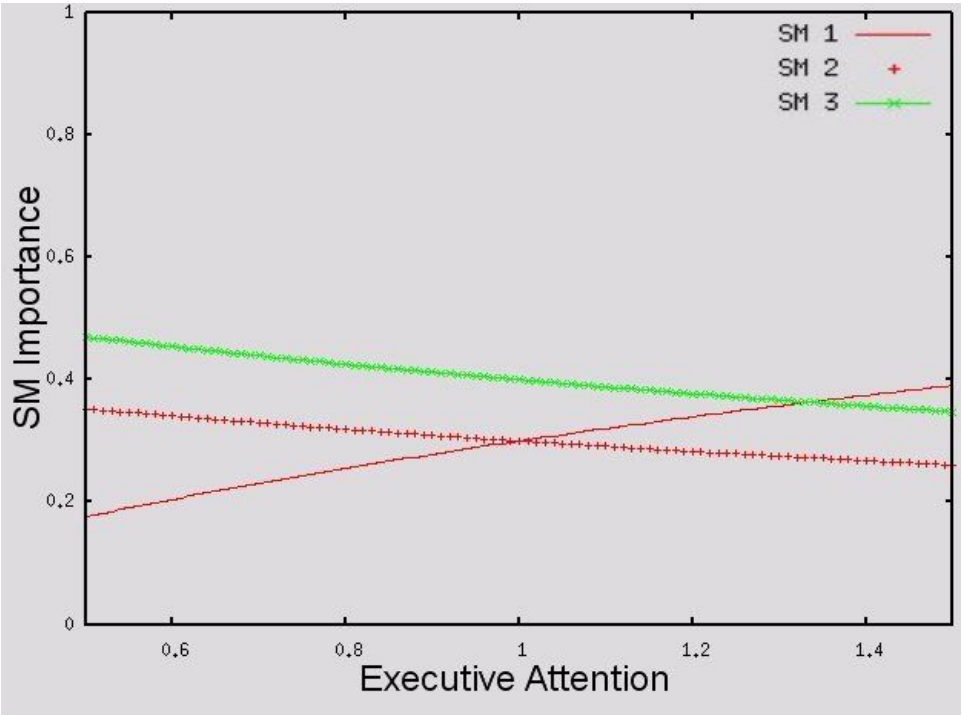

Figure 46. Effect of Executive Attention on SM Importance

Figure 46 shows the effect that attention from the Central Executive has on the importance of memory retrieval. In this case, the contextual importance of SM 1 and 2 was set to be .3, while the importance of SM 3 was .4. The effect of the executive attention was linear on the importance of a memory. This is an important observation,

because the executive attention has the same effect on SM importance as does the contextual relevance of the SM unit.

The importance of executive attention will be demonstrated through an analysis of the retrieval of a memory based on its relevance to the cognitive system.

An example cue for this experiment can be described as:

Table 16. Cue for EM Experiment 3

Goal	***Subject***	***S-WM***	***ST-WM***
Pick up	Blue Bag	bb, Pick up, Person 1	yb, rb, Person 1

This cue is derived from the description of the environment of the experiment as discussed earlier in Chapter VI. In the above table, rb represents the semantic memory node for the red bag, yb is the yellow bag, and bb is the blue bag. ISAC is performing the task of picking up an object (in this case the blue bag) that is not currently visible. ISAC must find the "lost" blue bag by retrieving a recently formed Blue Bag memory as discussed in Appendix B.

This cue is translated into the following histogram through the process shown in Figure 25.

Table 17. SM Frequency List for EM Experiment 3

SM Node	***% Constituent***
Pick up	25
Blue Bag	25
Person 1	25
Yellow Bag	12.5
Red Bag	12.5

After weighting for rarity and cognitive weighting, the fingerprint shown in Table 17 is created. The rarity of each SM node was calculated through the use of the Episodic Memory table listed in Appendix B.

Table 18. Unnormalized Fingerprint for EM Experiment 3

SM Node	***Score (% importance unnormalized)***
Pick up	25 * .25 * 1 = 6.25
Blue Bag	25 * .2 * 1.5 = 7.5
Person 1	25 * .2 * 1 = 5
Red Bag	12.5 * .25 * 1 = 3.125
Yellow Bag	12.5 * .25 * 1 = 3.125

After normalization, the final fingerprint is obtained.

Table 19. Normalized Fingerprint for EM Experiment 3

SM Node	***Score (% importance)***
Blue Bag	30
Pick up	25
Person 1	20
Red Bag	12.5
Yellow Bag	12.5

The Blue Bag SM unit is the most important to this scenario due to the cognitive weighting. The Pick up SM node is the next most important, while the Blue Bag is the most relevant SM unit that occurs alongside the “Pick up” goal. Tasks involving picking up the blue bag will therefore be the most likely to be retrieved in this scenario. The Pick

up the Blue Bag (PUBB) episode is contrasted with the recently formed Visually Track Blue Bag (VTBB) experiment.

The history term for a memory with an age of 1 is defined as 1. This is also shown in Figure 45.

Table 20. Calculation of Scores for EM Experiment 3

Episode	***Fingerprint***	***Occurrence***	***History***	***Score***
VTBB	0.300	0.250	1.000	0.075
PUBB (BB)	0.300	0.250	0.300	0.023
PUBB (PU)	0.250	0.125	0.300	0.009

The final score for the VTBB episode was .075. The score for the PUBB episode was determined to be .031. This means that the history factor for the VTBB episode could decay to a value of .413 before it relinquished the top slot in the working memory. This would take approximately .7 time units.

. A quick analysis can show if the situation would be different were executive attention not used.

Table 21. Unnormalized Fingerprint for EM Experiment 3 Without Executive Attention

SM Node	***Score (% importance unnormalized)***
Pick up	25 * .25 * 1 = 6.25
Blue Bag	25 * .2 * 1 = 5
Person 1	25 * .2 * 1 = 5
Red Bag	12.5 * .25 * 1 = 3.125
Yellow Bag	12.5 * .25 * 1 = 3.125

After normalization, the final fingerprint is obtained.

Table 22. Normalized Fingerprint for EM Experiment 3 Without Executive Attention

SM Node	***Score (% importance)***
Blue Bag	22
Pick up	28
Person 1	22
Red Bag	14
Yellow Bag	14

The Pick up task has now become the most important semantic memory unit. The Pick up the Blue Bag (PUBB) episode is once more contrasted with the recently formed Visually Track Blue Bag (VTBB) experiment.

Table 23. Calculation of EM Scores for EM Experiment 3 Without Executive Attention

Episode	***Fingerprint***	***Occurrence***	***History***	***Score***
VTBB	0.220	0.250	1.000	0.055
PUBB (BB)	0.220	0.250	0.300	0.017
PUBB (PU)	0.280	0.125	0.300	0.011

Once again, the VTBB episode remains in front, however it only would have to decay halfway to begin to fade. This process would take on the order of half a time unit, significantly less than the time it would have taken with executive attention factored in.

As in Experiment 2, this experiment was conducted under the conditions listed in Appendix B, and the target memory was retrieved successfully.

Episodic Memory Experiment 4

Finally, ISAC was asked to perform a similar movement task to one that it performed earlier. The cue was "Pick up the orange bag", a task that ISAC had never before performed, while the target episode was one that contained a different "pick up" experience, such as "Pick up the blue bag" or "Pick up the Barney". To retrieve this target episode that held little similarity to the current environment, ISAC had to use attention, which would be provided by the CEA in a fully integrated system, to enhance the importance of the task (see Chapter V for more information on this process).

Figure 46 shows the impact that the executive attention has on a SM unit when calculating the importance of that unit for retrieval. This influence is the same as that of contextual relevance, so both these factors combine to produce the correct retrieval in this experiment.

The cue for this experiment is very similar to that of EM Experiment 3, except an unknown Orange Bag is referenced.

Table 24. SM Frequency List for EM Experiment 4

SM Node	***% Constituent***
Pick up	25
Orange Bag	25
Person 1	25
Yellow Bag	12.5
Red Bag	12.5

After weighting for rarity and cognitive weighting, the fingerprint shown in Table 25 is created. Because the CEA is interested in retrieving an episode similar in task, the

Pick up goal is highlighted by cognitive attention. The rarity of each SM node was calculated through the use of the Episodic Memory table listed in Appendix B.

Table 25. Unnormalized Fingerprint for EM Experiment 4

SM Node	***Score (% importance unnormalized)***
Pick up	25 * .25 * 1.5 = 9.375
Orange Bag	25 * 1 * 1 = 25
Person 1	25 * .2 * 1 = 5
Red Bag	12.5 * .25 * 1 = 3.125
Yellow Bag	12.5 * .25 * 1 = 3.125

After normalization, the final fingerprint is obtained.

Table 26. Normalized Fingerprint for EM Experiment 2

SM Node	***Score (% importance)***
Orange Bag	55
Pick up	21
Person 1	10
Red Bag	7
Yellow Bag	7

It can be seen that the Pick up semantic memory unit is far more important than any of the others. The Orange Bag is excluded from retrieval because it is by definition not referenced by any episodic memory units that could be considered for retrieval. It is fairly obvious that a memory containing the task of Picking up the Red Bag would be retrieved given the corpus in Appendix B.

If there had been an episode containing information about the Orange Bag, its importance would have diminished greatly. The following fingerprint is generated with only two references to the Orange Bag:

Table 27. Fingerprint Generated with References to Orange Bag in EM Experiment 4

SM Node	***Score (% importance)***
Orange Bag	38
Pick up	28
Person 1	16
Red Bag	9
Yellow Bag	9

It can be seen that the difference between the Orange Bag and Pick up is not as drastic as in the first fingerprint generation, however the orange bag episodes have a strong chance of inclusion in the E-WM. This could cause failure of the retrieval system due to a relatively novel percept outweighing the task-related Pick up SM unit.

This could be combated through incremental increases in executive attention until the correct memory was retrieved. It is dangerous to automatically increase the executive attention to a high level upon directed memory query, however, because it prevents the E-WM from retrieving information that might be used for changing goals based on an environmental percept.

In this case, the goal SM unit and executive attention outweigh any of the factors outlined in the previous two examples, causing several episodes involving a “Pick Up” task to be retrieved. This experiment was successfully conducted using the preconditions listed in Appendix B.

EM Experiment Discussion

These three examples (EM Experiments 2-4) show how each of the elements that enters into the retrieval of a target EM unit combine to help perform the correct retrieval. Each of these situations is similar to one that ISAC could encounter during routine use, and the EM retrieval algorithm retrieves the correct memory in each case.

Several assumptions are made in the retrieval process. First, it is assumed that the more a memory constitutes an episode, the more important it is within the context of that episode. This assumption is valid because the stated purpose of the working memory system is to select those memories that are relevant to the current situation. A second assumption made by the retrieval process is that the emotion agent performs rationally with respect to the desired performance of the Episodic Memory, and generates salience values that make sense to task execution in ISAC's environment. If this assumption is invalid, the retrieval system will decay memories at rates that are not appropriate to their future importance to the cognitive system, and the Episodic Memory system will not function like it should.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

The creation of a memory system for a cognitive robot has been detailed. Several different types of memory systems have been outlined and implemented within the framework of ISAC's cognitive system.

A Procedural Memory was created to allow ISAC to store behaviors. This PM accepts processed behaviors from the ST-ISOMAP dimensionality reduction system, and stores them for later retrieval by the Modular Controller. Within the Modular Controller, the TD-Learning system attempts to predict the expected reward provided by each PM unit to the movement command, and loads the P-WM accordingly.

The Semantic Memory provides a way to represent semantic facts in a consistent manner throughout the system. A perceptual process was detailed that showed how the Semantic Memory could be used to recognize and represent objects. The S-WM is populated by direct search.

An Episodic Memory system was shown that recorded the state of the system during the execution of a task. A retrieval technique was developed for the EM system that allowed for the inclusion of several factors important for retrieval of the correct memory into the E-WM.

Although memory design is important, ultimately the performance of the system reduces to those acts that the system performs with the memories. With this in mind, several future research topics hold potential as a beneficial place to build from the memory systems.

The first is the adoption of the Working Memory Toolkit mentioned in Chapter 3 for use in the Modular Controller. This Toolkit is much more dynamic than the system currently implemented, and holds promise to help ISAC generalize over different motions instead of having to learn motions independently.

There are three main advantages to the implementation of the WM Toolkit to ISAC's Procedural Memory selection.

The first is that it can generalize between similar movement commands provided by the CEA so the system does not have to relearn which behaviors are appropriate for movements that are similar in function.

The second is that the WM Toolkit is able to judge PM units based on their characteristics. Similar PM units would be associated within this framework. In other words, characteristics of these PM units could serve to provide the system with a method of judging candidate PM units instead of the current method of blindly guessing which PM units are suitable.

Thirdly, the use of the WM Toolkit would also allow for easy expansion. Because memories and movement commands are judged based on characteristics, the addition of a new memory or movement command would not cause a disturbance in the system. If the expected reward provided by the system to a new PM unit were to be assigned randomly, as it currently is, the new unit might take several iterations to settle to a usable state.

Because of their extremely specialized nature, the other Long Term Memory Working Memory systems are not particularly suited to the application of the Working Memory Toolkit. However, should the attentional network of the SES not function as it should, the Working Memory Toolkit would be ideal for populating the Short Term Working Memory.

The main obstacle to this implementation is encoding the system state and candidate Semantic Memory units into appropriate feature vectors. Feature vectors that are nearly binary in nature must be generated from these candidate memories, and the aspects of the memories that should be used when comparing them with others for inclusion in the Short Term Working Memory need to be considered in detail when contemplating encoding methods.

A second area of future research is the creation of Semantic Memory units via perceptual processes. Although a sample perceptual module was presented, many must be developed to harness different computational methods of identifying objects. The perceptual process must act independently, and must be able to recognize novelty. The addition of any new perceptual process adds to the robustness of ISAC's perceptual system, and therefore to ISAC's interaction with the world.

Another area of work that is needed for the SM system is the inclusion of perceptual data from psychology with the vision research. Much work has been performed analyzing how humans classify objects based on various characteristics, and this work could be applied to a perceptual algorithm for classification that is superior to the Cartesian method described.

Finally, methods to translate Episodic Memory traces to schema for planning and control should be investigated. Episodes are essentially filled-in plans, methods must be developed for general extraction of schema from these records, and the execution of such schema for task solving. If this is accomplished, ISAC would be able to learn from example and build on experience.

Episodes could also be used to generate a more general problem solving strategy, in which plans are combined over time to generate permanent schema contained within

the CEA. This could be accomplished through employing a statistical method such as a Bayesian model or a Hidden Markov Model.

After the cognitive system as a whole is implemented, the various coefficients should be adjusted in order to maximize the performance of the memory systems within their new context. These coefficients include the time unit used for EM decay, the mapping from salience to the decay constant Alpha, and the mapping from emotion vector to emotional salience scalar. It is impossible to know what values will work in advance or what ISAC's environment is going to contain, so these values must be set when the system is integrated.

APPENDIX A

SOFTWARE USER AND IMPLEMENTATION GUIDE

This Appendix describes the correct use and configuration of the software, and details some of the database information needed in case of total data loss.

MySQL Database

The MySQL Database is designed to run under a single database. The following dump of the database will allow for contextual reference of the programs written in pursuance of this thesis as well as future recreation of the database upon loss of the backup files.

```
-- MySQL dump 9.11
--
-- Host: localhost    Database: isac
CREATE TABLE EM_Episodes (
  id bigint(20) unsigned NOT NULL auto_increment,
  startTime datetime default NULL,
  endTime datetime default NULL,
  accessTime datetime default NULL,
  goal text,
  STMLog text,
  DMLog text,
```

```
  EmotionLog text,
  HALog text,
  SALog text,
  AccessCount int(11) default NULL,
  alpha decimal(40,30) default NULL,
  Salience int(11) default NULL,
  Fingerprint text,
  History text,
  PRIMARY KEY  (id)
) TYPE=MyISAM;
CREATE TABLE EM_WMS (
  handle text,
  WMSize int(10) default NULL
) TYPE=MyISAM;
CREATE TABLE EM_fingerprint (
  fingerprint text
) TYPE=MyISAM;
CREATE TABLE PM_Behaviors (
  id int(11) NOT NULL default '0',
  name varchar(64) default NULL,
  lengthInMPs int(11) default NULL,
  MPs text,
  timestamp datetime default NULL,
  description text
```

```
) TYPE=MyISAM;
CREATE TABLE PM_MPs (
  id int(11) NOT NULL auto_increment,
  RawData text,
  ReverseDeps text,
  Statistics text,
  PRIMARY KEY  (id)
) TYPE=MyISAM;
CREATE TABLE PM_RawData (
  id int(11) NOT NULL default '0',
  data text,
  InstanceOf int(11) default NULL
) TYPE=MyISAM;
CREATE TABLE SA_WMS (
  handle text,
  emotion text,
  emotionScalar float default NULL,
  WMSize int(10) default NULL,
  FormingEMFingerprint text,
  FormingEM text,
  goal text,
  movement text,
  Focus text
) TYPE=MyISAM;
```

```
CREATE TABLE SES_WMS (
  handle text,
  WMSize int(10) default NULL
) TYPE=MyISAM;
CREATE TABLE SM_ExampleNode (
  id int(11) NOT NULL default '0',
  Modality varchar(32) default NULL,
  Data blob,
  InstanceOf int(11) default NULL
) TYPE=MyISAM;
CREATE TABLE SM_Objects (
  id bigint(20) unsigned NOT NULL auto_increment,
  Name varchar(32) default NULL,
  Type varchar(32) default NULL,
  RecInfo text,
  SampleData text,
  Characteristics text,
  Refs int(11) default NULL,
  Class varchar(32) default NULL,
  EM_hash text,
  PRIMARY KEY  (id)
) TYPE=MyISAM;
CREATE TABLE SM_RecNode (
  id int(11) NOT NULL default '0',
```

```
  InstanceOf int(11) default NULL,
  Method varchar(32) default NULL,
  Data text
) TYPE=MyISAM;
CREATE TABLE SM_WMS (
  handle text,
  WMSize int(10) default NULL
) TYPE=MyISAM;
```

Each Perl file that accesses the database contains the following code near the start of the file:

```
my $server = 'localhost';
my $db = 'DM';
my $username = 'username';
my $password = 'password';
```

Obviously, these values will have to be changed to fit the environment in which the program is to be run. Each Perl file also contains uses the DBI module, available from CPAN at http://www.cpan.org. Get your system administrator to install them, as root access is required. Note that the Perl files are location-independent and may be run at any location on the network as long as the MySQL database IP is known and the database is configured to allow remote connections.

PMComplianceChecker.perl

PMComplianceChecker parses the text files and ensures that they are compatible with the format. It will print diagnostics so the user can tell which data is being read incorrectly.

The program is called with the command "./PMComplianceChecker.perl <filename>".

The text files are in the format:

1 format0

2 Behavior2

3 .24 .32 .58

4 4 2

5 ./example1.txt

6 ./example2.txt

Line 1 allows for expansion of the parsing program for multiple formats. For example, with a change in the system, the value format1 could be specified.

Line 2 specifies the name of the behavior. This string is stored in the "name" field of the PM_Behaviors data field.

Line 3 tells the database where the motion primitives start and stop. For example, the first motion primitive runs from 0% to 24% of the example files, and the second motion primitive runs from 24% to 32% of the example files. This allows the example files to have different sampling times.

Line 4 tells the number of KeyTimes and the number of example files separated by a space. The number of KeyTimes is redundant with line 3, however it was included for compatibility with other programs.

Lines 5 and up list the filenames containing the example data. These files take on the following format:

1	3	7					
2	1	.24	.35	.387	.2034	.11934	.1694
3	2	.25	.35	.382	.2024	.140934	.1594
4	3	.27	.35	.381	.2044	.150934	.1494

Line 1 contains the number of rows and the number of columns in the data file separated by a tab character. The following lines contain the data, separated by tabs.

PMBehaviorMaker.perl

PMBehaviorMaker populates the database with the behaviors contained within the files on which it is called. It is called by the command "./PMBehaviorMaker.perl <filename>".

PMClearTables.perl

PMClearTables.perl clears the Procedural Memory Database of all records. **Do not run this program unless you would like to delete the entire database.**

Semantic Memory

The perceptual program contains two components: the C++ program that parses the camera images and generates the color ovuloids for the objects it finds, and the Perl program that compares the color ovuloids generated by the C++ program to those contained within the database for existing objects and determines if the detected object is novel. These two components are intended to be run from a script, which should be created according to the system running the perceptual processes.

The C++ program is run by the command "./Vision <imagefile.jpg>". The perl program is then run by the simple command "./PopulateSM.perl". This program will proceed along the perceptual process detailed in Chapter VII. There are immediate plans to port this program to run automatically with ISAC's cameras, more immediate incantations will be stored on the CIS server under the "Demos" directory.

Episodic Memory

The Episodic Memory daemon is split into three parts: the formation daemon, the EWM population daemon, and the EM decay script. All three programs are, while commented, extremely dense perl code, so care should be taken when editing, backups of versions should be made regularly, and helper functions (especially in the decay program) may be reused in the case that rewriting is preferable to debugging my code. Remember, "There's more than one way to do it."

The formation daemon is meant to be run continually during robot execution. It was not placed into a looping script because the execution cycle of the CEA is not known. For example, if the CEA executes goals sporadically, one does not want to record the periods between goals into an EM unit. Once the CEA is designed, perhaps a "NULL"

goal should be placed in the S-WM database and the formation daemon could poll until a different goal were placed in the database.

The formation daemon is called by the command: "./EM-Formation-Daemon.pl". It requires the installation of the Data::Dumper module from CPAN (see end of MySQL section).

The EWM population daemon should not harm anything by looping continuously. This could be accomplished by a simple bash script that sleeps between script calls, or its execution could be tied to the execution of the CEA by a method similar to the one described in the EM Formation script. The EWM population daemon is run by calling the file "./EM-WMS-Populate.pl".

The EM History update script should be run infrequently as it loops through the entire database, updating history values and pruning memories as is warranted. It should be called by a system cron job once an hour (or once a day if resources are valuable). The script currently uses days as its unit of time. To change this value, change the code "$dur->in_units('days');" to indicate your unit of choice. This code is contained in the middle of the file, a placement that could not be avoided due to the sequence of execution.

The history update script requires the CPAN modules DateTime and DateTime::Format::MySQL. To run the history update daemon, issue the command "./EM-History-Update.pl".

The program EMClearEpisodes.perl deletes all EM records in the database. Use at your own risk.

The three Octave files (EM-Analysis.m and EM-DBSize.m) are included within the codebase for future system analysis, and to provide implementations of the EM decay function in code that may be used for broader system analysis in the future. For example,

an analysis of the total space complexity of ISAC's cognitive system as a whole may be required in the future, and these files may be used to produce appropriate values for the EM.

ISAC Simulation Programs

Two programs were written to simulate simultaneously the operation of ISAC's WMS and CEA. These were written for use in creating Episodic Memory units for testing the EM and EWM, however they will be of use when building and debugging any application that requires memory access, such as the CEA or the Emotion Agent.

The first program, the ISAC cognitive simulator, reads all SM units from the memory database, and allows the user to move these units to the WMS and Self Agent WMS. It also allows the user to view the contents of the forming Episodic Memory unit.

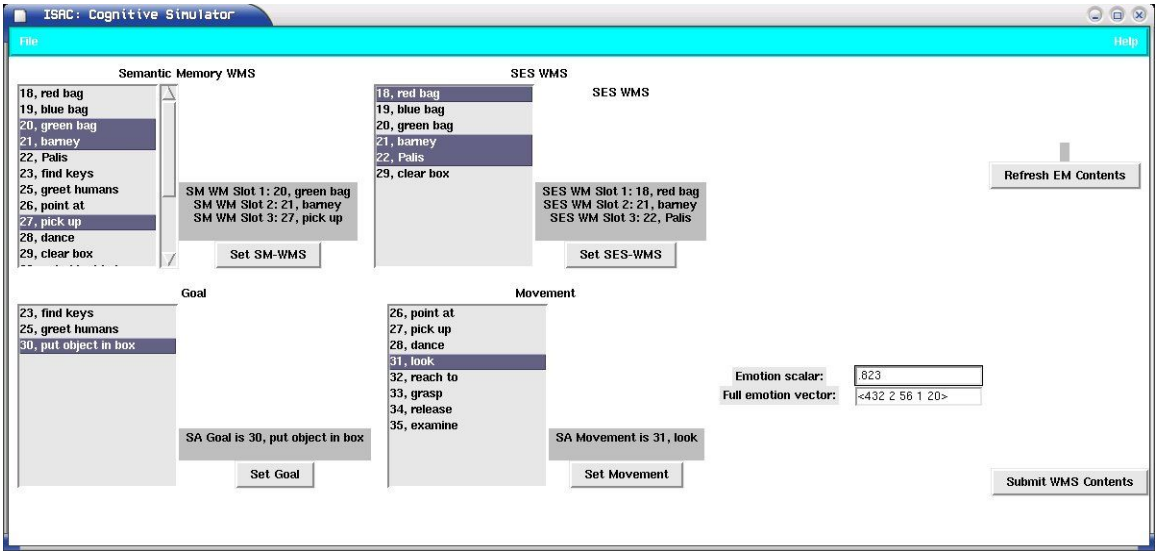

Figure 47. ISAC Cognitive Simulator

Figure 47 shows the user interface of this program. Appropriate SM units should be selected from the lists of available SM units. Upon selection of the appropriate units and entry of the emotion information, the "Submit WMS Contents" button should be pressed. If the user wishes to see the exact units that are highlighted, the button to the right of the lists may be pressed, however this merely displays user feedback and should be ignored if timing is required. This program (and the next) requires the installation of the Tk graphical toolkit and the Tk perl module from CPAN. The ISAC Cognitive Simulator may be started with the command "./ISAC-Simulator.pl".

The second program allows the user to easily create SM units. Although the information contained within the SM unit may not be manipulated within this program, it is useful for debugging the way in which memory systems pass around this information and use the description field of the SM unit.

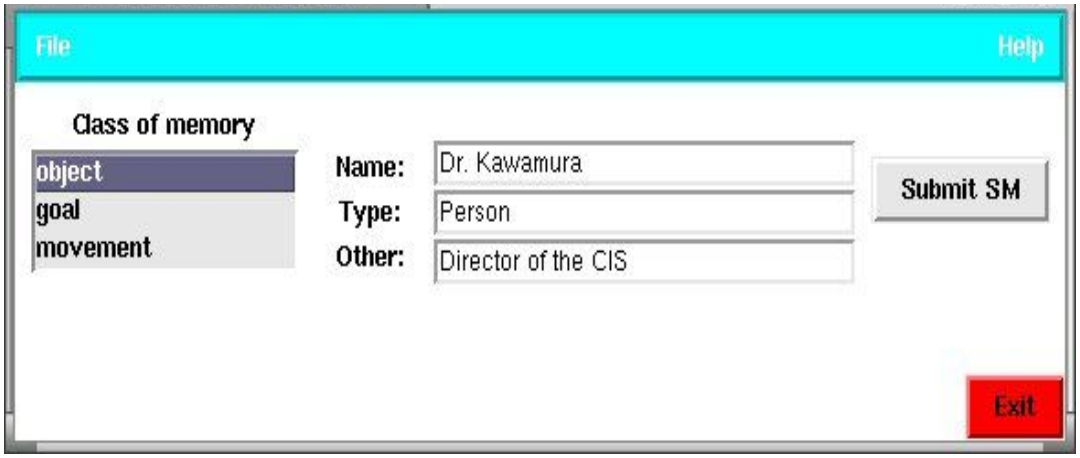

Figure 48. SM Creation Program

Figure 48 shows the interface of the SM creation program. The "Submit SM" button should be clicked when the fields are filled out correctly.

APPENDIX B

DETAILED EPISODIC MEMORY EXPERIMENTAL SETUP AND WALKTHROUGH

The Episodic Memory system was populated with 32 episodes, representing tasks that ISAC might encounter in normal operation:

Table 28. Contents of EM Database for EM Experiments

Goal	***Subject***	***Age***	***Salience***
Pick up	Red Bag	2	0.5
Pick up	Blue Bag	2	0.5
Pick up	Yellow Bag	2	0.5
Pick up	Barney Doll	2	0.5
Place in box	Red Bag	2	0.5
Place in box	Blue Bag	2	0.5
Place in box	Yellow Bag	2	0.5
Place in box	Barney Doll	2	0.5
Visually track	Red Bag	2	0.5
Visually track	Blue Bag	2	0.5
Visually track	Yellow Bag	2	0.5
Visually track	Barney Doll	2	0.5
Point to	Red Bag	2	0.5
Point to	Blue Bag	2	0.5
Point to	Yellow Bag	2	0.5
Point to	Barney Doll	2	0.5
Greet	Person 1	2	0.5
Greet	Person 2	2	0.5
Greet	Person 3	2	0.5
Greet	Person 4	2	0.5

Goal	*Subject*	*Age*	*Salience*
Handshake	Person 1	2	0.5
Handshake	Person 2	2	0.5
Handshake	Person 3	2	0.5
Handshake	Person 4	2	0.5
Visually track	Person 1	2	0.5
Visually track	Person 2	2	0.5
Visually track	Person 3	2	0.5
Visually track	Person 4	2	0.5
Wave at	Person 1	2	0.5
Wave at	Person 2	2	0.5
Wave at	Person 3	2	0.5
Wave at	Person 4	2	0.5

Figure 25 shows the formation process for Episodic Memory units in detail. This process must be simulated by passing links to Semantic Memories directly into the Episodic Memory. The simulation of the state of the cognitive system is performed by a simulator written in Perl with the graphics toolkit Tk. This program is pictured in Figure 47.

It is important to remember that each episode contains many Semantic Memory units in a sequence. For example, the Pick Up the Red Bag task might produce the following episode:

(bb = blue bag, rb = red bag, ba = barney, pu-rb = pick up red bag, r = reach, ch = close hand, la = lift arm, s = successful execution)

Table 29. Sample EM Unit Composition

t	*S-WM*	*S-WM*	*Movement*	*Goal*
1	bb, rb, ba	rb, pu-rb, r	r	pu-rb
2	bb, rb, ba	rb, pu-rb, ch	ch	pu-rb
3	bb, rb, ba	rb, pu-rb, la	la	pu-rb
4	bb, rb, ba	rb, pu-rb, s	s	pu-rb

The salience of the emotion rises throughout the episode because more subgoals are accomplished (such as having the red bag in hand, or holding the red bag over the table). The ST-WM would also have positional information for each unit, this was omitted from the table for clarity.

Two target EM units were then added to the database for use in the experiments as described in Chapter V. The two episodes are described as follows:

Table 30. Target EM Episodes Added to EM Database Contents for EM Experiments 2-4

Goal	*Subject*	*Age*	*Salience*
Visually track	Blue Bag	1	0.5
Visually track	Red Bag	3	1

The second described episode contains the “Fire!” stimulus and information. The original goal of this task was to visually track an object – learning the implications of the “Fire!” experiment took place within this episode after the stimulus was presented.

One cue was then created for each of the three scenarios. This cue is like one that would be seen at the start of the described task.

BIBLIOGRAPHY

Albus, J. S., "Outline for a theory of intelligence," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no.3, pp.473–509, 1991.

Alford, W. A., T. Rogers, D. M. Wilkes, and K. Kawamura, "Multi-Agent System for a Human-Friendly Robot", *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics (SMC '99)*, pp. 1064-1069, October 12-15, 1999, Tokyo, Japan.

Anderson, J. R., D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind", *Psychological Review 111,* (4). pp. 1036-1060, 2004.

Anderson, J., *The Adaptive Character of Thought*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1990.

Baars, B. J. and S. Franklin, "How conscious experience and working memory interact", *Trends Cogn. Sci. 7*, 166–172, 2003.

Baars, B. J., *A Cognitive Theory of Consciousness,* Cambridge University Press, 1988.

Baddeley, A., *Working Memory*, 11. Oxford Psychology Series. Oxford: Clarendon Press, 1986.

Benjamin, P., "Robotics Lab", http://csis.pace.edu/robotlab/ (accessed March 15, 2005).

Braver, T. S. and J. D. Cohen, "On the control of control: The role of dopamine in regulating prefrontal function and working memory", *Control of Cognitive Processes: Attention and Performance XVIII*, Eds. S. Monsell & J. Driver, pp. 713-738. MIT Press, 2000.

Brooks, R. A., "Intelligence without Representation", *Artificial Intelligence 47*, 139-159, 1991.

Budiu, R. and J. R. Anderson, "Verification of sentences containing anaphoric metaphors: An ACT-R computational model", In *Proceedings of the Fifth International Conference on Cognitive Modeling*, Eds. F. Detje, D. Doerner, & H. Schaub, pp. 39-44, Bamberg, Germany: Universitats-Verlag Bamberg, 2003.

Budiu, R., *ACT-R About*, http://act-r.psy.cmu.edu/about/ (accessed March 15, 2005).

Conway, M. A., *Flashbulb memories,* Hove, U.K.: Erlbaum, 1995

Cowan, N., "The magical number 4 in short-term memory: A reconsideration of mental storage capacity", *Behavioral and Brain Sciences*, 24(1): pp. 87–185, 2001.

Dodd, W. and R. Gutierrez, "The Role of Episodic Memory and Emotion in a Cognitive Robot", IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN), Nashville, TN, 2005.

DuBois, P., *MySQL, 2nd Edition*. New Riders Publishing, 2003.

Erol, D., J. Park, E. Turkay, K. Kawamura, O.C. Jenkins and M.J. Mataric, "Motion generation for humanoid robots with automatically derived behaviors," *Proc. of IEEE Int'l. Conf. on Systems, Man, and Cybernetics,* Washington, DC, Oct. 6-8, 2003, pp. 1816-1821, 2003.

Foresight Contributors, "Cognitive Systems: A Summary", http://www.foresight.gov.uk/Previous_Projects/Cognitive_Systems/Defining_the_Project/Cognitive_Systems__A_Summary.html (accessed March 15, 2005).

Franklin, S., *Artificial minds.* Cambridge, MA: MIT Press, 1995.

Franklin, S., "Autonomous Agents as Embodied AI," *Cybernetics and Systems Special Issue on Epistemological Aspects of Embodied AI*, 28:6 499-520, 1997.

Franklin, S. and A. Graesser, "Is It an Agent, or Just a Program? A Taxonomy for Autonomous Agents," *Intelligent Agents III: Agent Theories Architectures, and Languages*, J. Mueller, Ed., Berlin: Springer, 1997.

Franklin, S., A. Kelemen, and L. McCauley, "IDA: A Cognitive Agent Architecture", *1998 IEEE Conf on Systems, Man and Cybernetics,* IEEE Press, 1998.

Franklin, S., "Action Selection and Language Generation in "Conscious" Software Agents", *Proc. Workshop on Behavior Planning for Life-Like Characters and Avatars*, i3 Spring Days '99, Sitges, Spain, 1999.

Franklin, S., and A. Graesser, "Modelling Cognition with Software Agents", *CogSci2001: Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, Ed. J. D. Moore, and K. Stenning. Mahwah, NJ: Lawrence Erlbaum Associates; August 1-4, 2001.

Friedland, N., P. Allen, G. Matthews, M. Witbrock et al., "Project Halo: Towards a Digital Aristotle", *AI Magazine*, 25:4, pp. 29-48, Winter 2004.

Fulton, J. and J. Pransky, "DARPA Grand Challenge – A pioneering event for autonomous robotic ground vehicles", *Industrial Robot: An International Journal*, 31:5, pp. 414-422, May 2004.

Fuster J.M., "Cortical dynamics of memory", *Int J Psychophysiol.* 35:155-64, 2000.

Goldman-Rakic, P.S., “Circuitry of the prefrontal cortex and the regulation of behavior by representational knowledge”, *Handbook of Physiology*, Eds. F. Plum and V. Mountcastle, pp. 373–417, Bethesda, MD: American Physiological Society, 1987.

Haikonen, P., “An Artificial Cognitive Neural System Based on a Novel Neuron Structure and a Reentrant Modular Architecture with Implications to Machine Consciousness”, Ph.D. Dissertation, Helsinki University of Technology, November 1999.

Haikonen, P., “An Artificial Mind via Cognitive Modular Neural Architecture” , *Proceedings Symposium on How to Design a Functioning Mind AISB00 Convention*, A.Sloman et al., Ed., 2000a.

Haikonen, P., “A Modular Neural System for Machine Cognition”, *Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks*, pp. 1047-1052, July 24 - 27, 2000b.

Haikonen, P., *The Cognitive Approach to Conscious Machines*. Exeter, UK: Imprint Academic, 2003.

Hambuchen, K.A., “Multi-Modal Attention and Binding using a Sensory EgoSphere”, Ph.D. Dissertation, Nashville, TN: Vanderbilt University, May 2004.

International Encyclopedia of the Social and Behavioral Sciences. Amsterdam: Pergamon (Elsevier Science).

Jenkins, O. C., and M. J. Matarić, “Automated derivation of behavior vocabularies for autonomous humanoid motion,” *2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, 2003.

Kanerva, P., *Sparse Distributed Memory*, Cambridge, MA: MIT Press, 1988.

Kawamura, K., R.A. Peters II, S. Bagchi, M. Iskarous, and M. Bishay, "Intelligent robotic systems in service of the disabled", *IEEE Transactions on Rehabilitation Engineering*, vol. 3, no. 1, pp. 14-21, March, 1995.

Kawamura, K., R. A. Peters II, D. M. Wilkes, W. A. Alford, and T. E. Rogers, “ISAC: Foundations in Human-Humanoid Interaction,” *IEEE Intelligent Systems,* July/August 2000, pp. 38-45, 2000.

Kawamura, K., W. Dodd, and P. Ratanaswasd, Robotic Body-Mind Integration: Next Grand Challenge in Robotics, Invited Paper, *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, Kurashiki, Japan, September 20-24, 2004.

Kawamura, K., W. Dodd, P. Ratanaswasd and R. A. Gutierrez, “Development of a Robot with a Sense of Self”, proceedings of *6th IEEE CIRA Symposium, Espoo, Finland, June 2005.*

Laird, J. E. and C. B. Congdon, *The Soar User's Manual Version 8.5 Draft*, 2004.

Laird, J. E., A. Newell, and P. S. Rosenbloom, "SOAR: An Architecture for General Intelligence", *Artificial Intelligence 33,* pp. 1-64, 1987.

Leake, D., "Case-based reasoning: experiences, lessons, and future directions", Menlo Park, CA: AAAI Press., 1996.

Mataric, M. J., "Studying the Role of Embodiment in Cognition", *Cybernetics and Systems*, 28(6), pp. 457-470, 1997.

McClelland, J. L., B. L. McNaughton, and R. C. O'Reilly, "Why There are Complementary Learning Systems in the Hippocampus and Neocortex: Insights from the Successes and Failures of Connectionist Models of Learning and Memory," *Psychological Review, 102,* pp. 419-457, 1995.

Minsky, M., *The Society of Mind*, New York: Simon and Schuster, 1985.

Norman, K. A. and R. C. O'Reilly, "Modeling hippocampal and neocortical contributions to recognition memory: A complementary learning systems approach," Psychological Review, Vol 110, pp. 611-646, Oct. 2003.

Nuxoll, A. and J. Laird, "A Cognitive Model of Episodic Memory Integrated With a General Cognitive Architecture," International Conference on Cognitive Modeling, 2004.

Pavlik, P. I. and J. R. Anderson, "An ACT-R model of memory applied to finding the optimal schedule of practice", In *Proceedings of the Sixth International Conference on Cognitive Modeling,* pp. 376-377, Pittsburgh, PA: Carnegie Mellon University/University of Pittsburgh, 2004.

Phillips, J. L. and D. C. Noelle, "A biologically inspired working memory framework for robots", In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pp. 1750-1755, Stresa, Italy, 2005.

Rainer, G., W. F. Asaad, and E. K. Miller, "Selective representation of relevant information by neurons in the primate prefrontal cortex", *Nature*, 393: 577–579, 1998.

Ramamurthy, U., M. Bogner, and S. Franklin, "Conscious Learning In An Adaptive Software Agent", *Proceedings of The Second Asia Pacific Conference on Simulated Evolution and Learning,* pp. 24-27, Canberra, Australia, 1998.

Ratanaswasd, P., S. Gordon, and W. Dodd, "Cognitive Control for Robot Task Execution", *Proceedings of 14th Annual IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN 2005)*, Nashville, TN, August 13-15, 2005.

Roberto E. Olivares, “The Intelligent machine architecture version 2.5: a revised development environment and software architecture”, Master's Thesis, Vanderbilt University, May 2003.

Rogers, T. E., “The Human Agent: A Model for Human-Robot Interaction”, Ph.D. Dissertation, Vanderbilt University, August 2003.

Rose, C., M.F. Cohen, and B. Bodenheimer, “Verbs and adverbs: Multidimensional motion interpolation”, *IEEE Computer Graphics and Appl.,* vol. 18, no. 5, Sept.-Oct. 1998, pp. 32-40, 1998.

Skubic, M., D. Noelle, M. Wilkes, K. Kawamura, and J.M. Keller, A Biologically Inspired Adaptive Working Memory for Robots, Accepted Paper, *2004 AAAI Symposium Series*, Washington, D.C., October 21-24, 2004.

Tulving, E., “How Many Memory Systems Are There?”, *American Psychologist*, 40, pp. 385-398, 1985.

Wray, R., R. Chong, J. Phillips, S. Rogers, and B. Walsh, “A Survey of Cognitive and Agent Architectures”, http://ai.eecs.umich.edu/cogarch0/ (accessed March 15, 2005).