HUMAN-INSPIRED FORGETTING FOR ROBOTIC SYSTEMS

By

Sanford T. Freedman

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

December 2010

Nashville, Tennessee

Approved:

Professor Julie A. Adams

Professor Gautam Biswas

Professor Douglas H. Fisher

Professor Gordon D. Logan

Professor D. Mitchell Wilkes

ACKNOWLEDGMENTS

This dissertation was supported by many individuals...

I was fortunate to have Dr. Fisher, Dr. Biswas, Dr. Logan, Dr. Wilkes, and Dr. Adams on my dissertation committee, a collection of exceptional researchers with diverse backgrounds. Each tackles problems with a different approach and despite being established figures in their respective fields, are eager to spend time assisting their students. Their advice and suggestions greatly improved the quality of my research, while providing the motivation to perform all of the experiments and write the dissertation.

My understanding of WiFi signals and the effects of various objects in the environment was the result of Dr Yuan Xue generously spending time discussing WiFi theory. Dr. Greg Walker spent many hours discussing the best approaches to take in optimizing the forgetting algorithms presented in this dissertation and many of the finer points related to the DAKOTA software package. I spent many hours in Andy Richter's office discussing the Vanderbilt University Electrical Engineering & Computer Science department's network infrastructure. These discussions made it possible to maximize the efficiency of my distributed software applications.

The collection of the real world results presented in this dissertation would not have been possible without the assistance of Sean Hayes. He spent an entire day manipulating equipment and logging data at the Colson Hollow group camp grounds. Eli Hooten and Stephen Motter provided their support during the construction of equipment required for the real world testing experiment. The real world experiment was fraught with delays and challenges, but Gary Hawkings was always willing to reschedule my experiment and provide the needed assistance to make the experiment possible.

This dissertation was partially supported by the Office of Naval Research MURI award N000140710749.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMES

ACT-R	The Adaptive Control of Thought - Rational
AI	Artificial Intelligence
CBRNE	Chemical, Biological, Radiological, Nuclear, Explosive
CLARION	Connectionist Learning with Adaptive Rule Induction ON-line
COGNET	COGnition as a NEtwork of Tasks
CoSy	Cognitive Systems for Cognitive Assistants
DAKOTA	The Design Analysis Kit for Optimization and Terascale Applications toolkit
H-CogAff	The Human-CognitionAffect Architecture
HRI	Human-Robot Interaction
PDF	Probability Density Function
SA	Situational Awareness
SIMPLE	The Scale-Independent Memory, Perception, and Learning Model
SNR	Signal to Noise Ratio
SSD	Signed Squared Deviation

CHAPTER I

Problem Statement

Perfect memory and recall provides a mixed blessing. While flawless recollection of episodic data and procedural rules allows for increased reasoning, photographic memory hinders a robot's ability to operate in real-time, highly dynamic environments. The absence of forgetting can result in memory being filled by a tremendous volume of data, increasing both search time and the probability of over-learning. Many small, but critical details within the environment greatly impact the probability of successful task completion, unfortunately robots are currently ill-equipped to navigate incoming data to detect, recognize, and act upon these details. As robotic hardware and designs improve, robots will be further inundated as finer resolution environmental data and higher accuracy mental models become available. Contemporary robots are already overrun with vast volumes of data requiring real-time processing and the problem will only increase. Before robots realize human-level intelligence, a means of classifying the importance of each acquired datum and forgetting unnecessary, erroneous, and expired data will be required. This dissertation uses forgetting to filter data and has developed a robotic forgetting mechanism, inspired by human forgetting, that incorporates time-based decay and interference effects.

Solomon Veniaminovich Shereshevskii, a professional mnemonist, possessed a close to photographic memory, but found it difficult to learn and understand higher level concepts. Despite the ability to recall lists of great length, understanding non-literal sentences or recognizing dynamically changing objects, including human faces, proved challenging (Luria, 1968). Shereshevskii's near photographic memory prevented him from forgetting minor details and allowing concepts to transcend into higher levels of thought (Schooler and Hertwig, 2005). A lack of forgetting capabilities can cause similar problems for robots.

Forgetting has previously been recognized as beneficial to machine-learning, particularly case-based reasoning (Kira and Arkin, 2004). Within some domains and tasks, random forgetting has been found to improve performance (Kira and Arkin, 2004) and in particular situations, even accurate data can hinder performance (Markovitch and Scott, 1988). Selective utilization of learned information has previously been used to prune classification hierarchies, improving efficiency (Yoo and Fisher, 1991; Markovitch and Scott, 1989; Mooney, 1989). Forgetting may be required in order to achieve human levels of intelligence, and theories of human forgetting may provide the basis for the generation of effective robot forgetting.

Through the years, strong evidence has been found to both support and refute the two leading theories of human forgetting, time-based decay and interference (Jonides et al., 2008). Some research has posited that human forgetting is a combination of both theories and their subcomponents (Jonides et al., 2008; Wixted, 2004; Sims and Gray, 2004; Altmann and Gray, 2002). Reflecting on the complexity of the human mind and cognition, it is unlikely that the full dynamic of human forgetting can be realized from one parsimonious mechanism. Several researchers have developed models of human forgetting that incorporate multiple forgetting methods (Mueller and Krawitz, 2009). Others that stand by one mechanism, have admitted that multiple mechanisms may contribute to human forgetting, just to a lesser extent (Wixted, 2004).

This dissertation uses forgetting as a means of filtering data and presents a new approach to implementing forgetting within robots that combines both time-based decay and interference theories of forgetting. Through the unique collection of mechanisms in the forgetting system, the approach may be highly amenable to the varied demands of robotic systems, allowing for the removal of irrelevant, erroneous, and out-dated information. Contemporary and future robotic systems involve a complex interaction between their various subcomponents. Individual robot subsystems may possess detailed and vital information regarding active tasks, but when this information is combined to form higher level concepts, a robot may be better equipped to detect and beneficially act upon the many critical details within the environment. The developed forgetting method is capable of simultaneously operating both at the lower raw sensor value levels, and with multiple instances of the forgetting mechanism, manage a robot's composite data. Inherently non-domain specific, the forgetting mechanism can operate on a wide array of data modalities, but can be parameter-tuned, allowing the relative importance of each component of the forgetting method to be modified.

Forgetting may directly aid the ability of robots to operate under uncertainty. When complete certainty is unavailable, then any decision or action selection made by a robot is essentially an educated guess. Heuristics comprise a critical form of decision making under uncertainty and real-time constraints (Gigerenzer et al., 1999; Gigerenzer and Brighton, 2009). Forgetting has previously been shown to improve the accuracy and applicability of some heuristics (Schooler and Hertwig, 2005). This form of decision-making benefits many areas within robotics and forgetting may significantly improve their effectiveness.

Neurobiology, psychology, artificial intelligence, and robotics have all worked cooperatively over the past few decades to improve the understanding of their respective fields. The forgetting approach presented in this dissertation may provide additional evidence for theories of human forgetting. The approach incorporates a large number of theories of human forgetting, but the impact of individual components can be independently modified, allowing for a comparison of the effects of each theory. As robots evolve and are assigned tasks with ever increasing similarity to the challenges faced by humans on a daily basis, the effects of individual forgetting components on robot performance may correlate with the effects of human performance.

Inherent to the operation of the dissertation's approach to robotic forgetting, a relevance metric is developed for each datum stored in a robot's knowledge bases. These values are used to classify and rank the probability that an individual datum will be required by the system. This classification may provide an additional search criterion for other subsystems within the robot.

The presented forgetting mechanism may indirectly aid robots in attaining human levels of Situational Awareness (SA). A phenomena with numerous definitions (Adams et al., 1995), SA can loosely be considered an understanding of the environment and effects of possible future actions. Data management poses one of the underlying difficulties in developing SA in real-time domains. Many features are required by a robot's architecture in order for comprehensive SA to be developed, such as *information filtering*, *inter-module communication*, and *storing volumes of information and purging stale data* (Freedman and Adams, 2008). These features all contain aspects of data management and are negatively affected by increasing volumes of stored data. Forgetting may aid the realization of these features and consequently improve the ability for robotic systems to develop SA.

CHAPTER II

Literature Review

Numerous factors can limit a robot's ability to effectively and efficiently operate within complex, dynamic environments. Copious volumes of data can inundate a robot's cognitive processes, information inaccessibility and incorrect knowledge can induce erroneous models of the environment, and demanding cognitive processes can prevent real-time responses to environmental stimuli. As robots increase their ability to assuage these data problems, real-time robotic performance may increase. Capabilities to forget undesirable data may aid robots attempting to minimize cognitive workload, prioritize data, and purge erroneous information.

This chapter provides a review of several aspects that are critical to the efficacious incorporation of forgetting mechanisms into contemporary robotic design. A history of robotics is presented to provide motivation and justification for taking inspiration from biological systems. Transitioning from the discussion of how biologically inspired ideas have influenced the evolution of robotics, human situational awareness (an understanding of the environment and effects of possible future actions) is discussed. Human situational awareness, a phenomena of key importance, that enables humans to effectively operate within challenging domains, may provide parallels allowing robots to operate in those same demanding conditions. Providing insight into the manner in which contemporary robots and cognitive agents are controlled, a review of several leading cognitive architectures is presented. Forgetting as occurs within humans is then reviewed. Human forgetting is a complex and multifaceted mechanism that offers many insights and suggestions.

II.1 History of Robotics

The field of robotics has a long and storied history, dating back thousands of years. The concept of robots originated as clever, sophisticated machines (Matarić, 2007), but slowly evolved into the myriad of present day definitions of the term robot (Wikipedia, 2009). These modern day definitions have refined the concept of the robot to roughly refer to intelligent machines possessing some level of autonomy. Originally mechanical creations generally possessing fixed modes of operation and minimal levels of adjustability, contemporary robotic designs have evolved into highly reconfigurable, general purpose machines (Wikipedia, 2009).

The term "Robot", coined by Josef Čapek, first appeared in the 1920 play Rossum's Universal Robots (R.U.R.) by Čapek's bother, Karel Čapek (Zunt, 2004); however, visions of mechanical beings performing undesirable, but necessary tasks had already existed for thousands of years (Matarić, 2007). Many examples of ancient robotic designs and technologies exist. Around approximately 1400 B.C., the Babylonians crafted a water clock named "Clepsydra", a device considered one of the earliest robotic devices (The History of

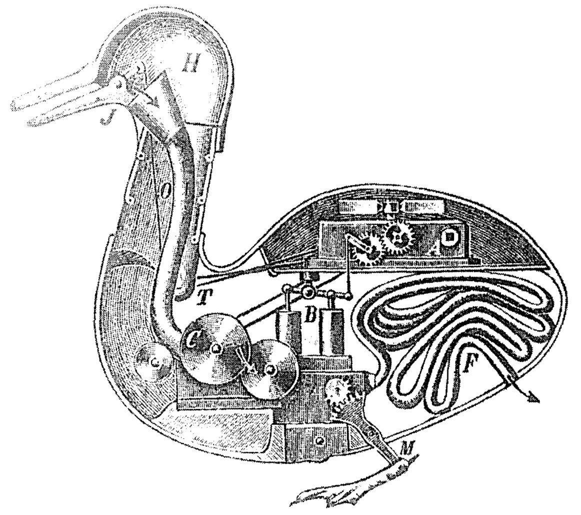


FIGURE 6. A nineteenth-century inventor's illustration of his own imagined version of a mechanical digesting duck. An arrow helpfully indicates where the main action takes place. From Chapuis and Édouard Gélis, *Le Monde des automates,* 2:151.

Figure II.1: Vaucanson's duck (Riskin, 2003)

Computing Project, 2007). Many user-configurable automated devices were crafted by Hero of Alexandria (10 - 70 A.D.), while a clock tower featuring mechanical figurines playing music to announce the passage of time was constructed in China by Su Song in 1088 (Wikipedia, 2009).

These early robotic creations realized tremendous mechanical discovery and ingenuity, but only reproduced the outward appearance and actions of the biological entities they mimicked. It was not until the eighteenth century that robotic devices began to reproduce both internal and external functions of the entities that they embodied (Riskin, 2003). In 1738, the French scientist Jacques Vaucanson put on display a mechanic duck, Figure II.1, that "stretches out its Neck to take Corn out of your hand; it swallows it, digests it, and discharges it digested by the usual Passage" (Vaucanson, 1979). Vaucanson's creation benefited from extensive analysis of real ducks to allow the mechanical creation to simulate a number of authentic behaviors and actions, although its most prominent, digestion, was faked (Riskin, 2003). This mechanical achievement was one of the earliest examples of robotic designs shifting from "amusements and feats of technological virtuosity" to the study of how biological processes could be recreated within machines and how the understanding of those processes may be enhanced (Riskin, 2003).

After Vaucanson created his duck, scientists and inventors continued developing devices of increasing realism and complexity, but it was not until the mid-twentieth century that robots began to resemble their modern day counterparts. In the 1940's, the field of cybernetics emerged, applying control theory to the understanding of biological systems. Cybernetics's initiator, Norbert Wiener and his colleagues investigated how control theory could explain and aid in simulating biological processes from the level of neurons to behavior (Matarić, 2007). During this time, William Grey Water used the field of cybernetics to develop a collection of devices commonly referred to as “tortoises” that exhibited many aspects analogous to biological systems. Many consider these turtles to be the first examples of modern robots (Matarić, 2007).

From the 1950’s through the mid-1980’s, robot developers started to integrate Artificial Intelligence (AI) inspired approaches into their creations. Robots such as Shakey (Raphael, 1976), HILARE (Giralt et al., 1979), and the Stanford Cart (Moravec, 1990) combined the biologically inspired abilities of vision processing, world modeling, and planning to achieve tasks too complex and difficult for earlier designs (Matarić, 2007). These robots achieved their success by employing the design paradigm commonly referred to as Sense-Model-Plan-Act. A robot using this approach first sensed its environment, and once complete, constructed a world model suitable for a planner. Plans were generated by this planner and subsequently transfered to an actuation unit that caused the robot to take physical actions (Brooks, 1991). Robots from this era often spent a majority of their processing time perceiving the environment and constructing world models (Brooks, 1991). In 1987, Schoppers (1987) developed the notion of Universal Plans, a lookup-table based approach to planning that required responses to all possible environmental states to be calculated.

Robots embodying the Sense-Model-Plan-Act paradigm were slow and required carefully engineered static environments (Matarić, 2007; Brooks, 1991). In the late 1980’s and early 1990’s, a new approach to robot design emerged, reactive (Brooks, 1986) and later behavior-based (Arkin, 1998; Matarić, 1992; Werger, 1999) robotics. Reactive design methods, analogous to human reflexes, enabled robots to respond to elements within the environment quickly, and unlike previous robotic designs, in a smooth manner (Matarić, 2007). Behavior-based robotics uses collections of simple modules that each exhibit individual behaviors. Collections of these behaviors run concurrently, resulting in emergent intelligence. Inspired by the capabilities of fish, insects, and birds, robots were developed with complex combinations of simple behaviors that resulted in abilities far exceeding earlier robot designs. A number of robots modeling insects were created, that in many ways, reproduced biological behavior (Arkin, 1998).

Despite minimalism and statelessness being core ideals of behavior-based robotics, researchers reintroduced deliberative processing into robotic designs. These new architectures were hybrid combinations of behavior-based principles and classical AI inspired algorithms (Werger, 1999). This change in design philosophy was inspired by the desire to realize higher level cognitive functions within robots. Present day robotic designs commonly employ this hybrid approach, often involving slow and highly computationally expensive processes operating on top of fast reactive mechanisms.

The reintroduction of state information into robotic designs allows robots to complete challenging cognitive tasks, but has proven to be a double-edged sword. Along with the benefits of persistent sensor data and mental constructs, enduring state can become stale and erroneous information may remain available to cognitive processes for significant periods of time. Humans remediate this informational plague by many means, one of which is forgetting. As future robotic designs progressively incorporate biologically inspired capabilities, parallels to human forgetting may allow robots to achieve still greater levels of performance and success.

II.2 Situational Awareness

Humans are able to operate and excel in highly dynamic and complex environments partly because of their perceptual and cognitive capabilities. Situation Awareness (SA) (Adams et al., 1995; Endsley, 1988b,a, 1995b), the human ability to perceive the environment, comprehend the situation, and project that comprehension into the near future, forms a critical component of those cognitive and perceptual capabilities.

SA has received considerable attention in the human factors community and many definitions have been developed (Uhlarik and Comerford, 2002; Fracker, 1988; Dominguez, 1994; Endsley, 1988b,a, 1995b; Sarter and Woods, 1991). As a result of SA's origin in avionics and military related human performance research, many existing definitions of SA are domain and task specific, such as SA

> "means that the pilot has an integrated understanding of factors that will contribute to safe flying of the aircraft under normal or non-normal conditions. The broader this knowledge is, the greater the degree of situational awareness" (Regal et al., 1988).

This dissertation will adopt the non-domain specific commonly accepted definition of SA from Endsley (1988b):

> "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future."

The large number of SA definitions presently available partly arose from the difficulty in determining exactly what is and what is not encompassed by SA (Uhlarik and Comerford, 2002). There has even been debate as to whether SA is a concept that defines a cognitive phenomena or if SA is a causal agent forming a portion of human cognition (Flach, 1995).

While often misinterpreted as a process (or a set of processes), SA is a grouping and description of processes working to form human cognition. The portions of these processes that handle incoming data and then compute higher level constructs can be grouped together by SA. Endsley partitioned this higher level construct into a series of three levels (Endsley, 1995b). Level one involves acquiring raw information from the environment. Level two consists of merging data from level one into an understanding of the current

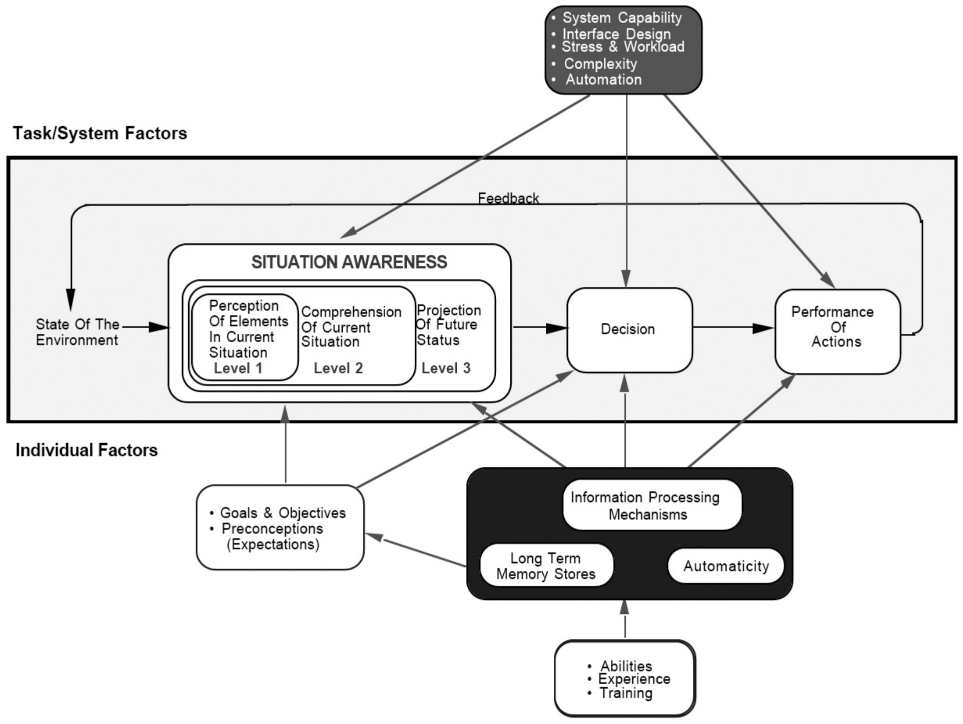


Figure II.2: Endsley's model of SA located within the greater model of human decision making (Endsley, 1995b)

situation in the environment. The final level utilizes level one and level two constructs to predict the future states of the environment and possible actions to undertake. Figure II.2 illustrates how Endsley's model of SA fits within the larger context of human decision making.

The concept of situational awareness can also be represented by the Perceptual Cycle (Neisser, 1976), a cyclical model of human cognition. As illustrated in Figure II.3, the actual environment modifies the human's current mental model of the environment's status and properties. These modifications result in active adjustments to how perceptual channels are directed within the environment. Perceptual channels sample the constantly changing environment, completing the cycle (Adams et al., 1995). While at a high level, the cyclical nature of situational awareness can be considered similar to Sense-Model-Plan-Act loops of robotics from the 1960's to the 1980's (Brooks, 1991), the various components of the perception cycle are not system states that can be discretely entered and exited. The three elements of the perception cycle are permanently active, simultaneously working toward high levels of situational awareness and performance.

The human cognitive system is highly advanced and at present too difficult to fully understand. The study of SA bounds the decision making process and aids in the identification of aspects of human cognition that contain similar properties and requirements (Flach, 1995). Care must be taken while studying SA as the

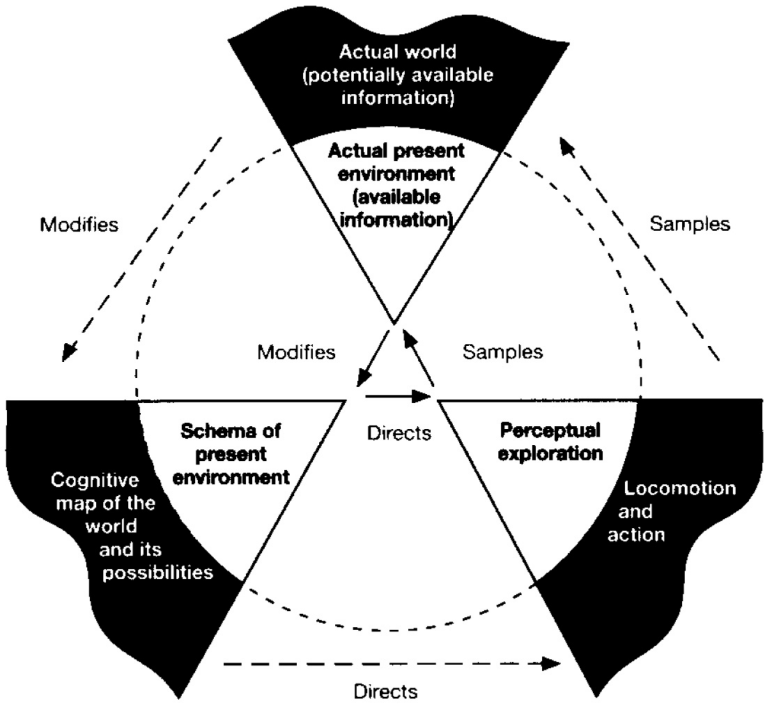

Actual world (potentially available information)
Actual present environment (available information)
Modifies
Samples
Modifies
Samples
Directs
Schema of present environment
Cognitive map of the world and its possibilities
Perceptual exploration
Locomotion and action
Directs

Figure II.3: The Perceptual Cycle (Neisser, 1976)

phenomenon is not performance, although a relationship between SA and performance exists (Adams et al., 1995). When minimal or erroneous SA is maintained, the likelihood of achieving high levels of performance is low, especially if a human mistakenly overestimates their level of SA. Inferior performance is not guaranteed as a result of poor SA, since luck, physical skill, and physical dexterity may compensate. If a human realizes that their level of SA is below some desirable threshold, then behaviors and actions may be changed to improve SA and causally improve performance (Endsley, 1995b). When an agent deliberately changes behaviors to improve SA, some means of approximating their SA level is required (New York Times News Service, 2000). In many domains, the ability to accurately estimate one's current level of SA may be critical to achieving high performance levels.

II.2.1 Situational Awareness Components

SA is difficult to define and isolate partly due to the large and diverse array of cognitive and perceptual abilities that compose the phenomenon, see Table II.1 for an initial list. Additionally, human SA is highly sensitive to a number of external (e.g. environmental stressors, salience, automation) and internal (e.g. cognitive workload, vigilance, fatigue, stress) factors (Endsley, 1995a; Salmon et al., 2006; Wickens, 2002). This section presents an overview of some of the cognitive and perceptual components that compose human SA.

Table II.1: Initial list of situational awareness components

Attention	Biases and Expectations
Explicit Focus	Schema and Mental Models
Short & Long Term Memory	Data Driven and Goal Driven Behavior
System Stress	Goals, Plans, and their Evolution
Workload	Observation Quality and Availability
Failures	Framing and Ordering Effects
Uncertainty and Confidence	Bad SA vs. Bad Decision Making
Ideal, Achievable, and Actual SA	Vigilance

II.2.1.1 Attention

Attention management affects a human's ability to sense and understand the surrounding environment. While attending to a particular sensory channel does not guarantee perception, attention is required to attain perception (Wickens et al., 2004), a critical component of level one SA. Humans are able to divide, direct, and select their attentional capabilities (Wickens and Hollands, 2000). However, human perception is limited and finite as humans have a bounded quantity of attention to distribute across all items of interest at any given point in time. These items generally consist of perceiving information from the environment, processing that information, decision making, and motor control. Human attentional resources are limited by sensory channel demands; and complex dynamic environments can quickly overload human attentive abilities (Freedman and Adams, 2007). As a result, humans may selectively sample their sensory channels (Smart, 2005). Humans typically manage their attentional focus based upon the frequency that percepts must be updated and information update rates. Sampling can occur in periodic sequences, but previously acquired information, along with current goals and plans, may also dictate the flow of attention. Human perception is further constrained by the limited human ability to parallel process sensor percepts due to sensor modality and working memory constraints. However, salient features may direct attention and focus their efforts to potentially critical tasks. As the presence of sampling increases, human SA can improve even as the domain becomes more challenging (Wickens and Hollands, 2000). Along with goal directed influences, the focus of attention can be shifted by expertise and ancestral bias (New et al., 2007).

Many domains are dynamic and information importance may change substantially throughout the duration of a task. Attention must shift and sampling patterns must be altered when these conditions occur in order to ensure incoming percepts are relevant and beneficial. Memory's collection of existing percepts may need to undergo a similar shift. When the applicability of individual percepts and memory items fade, forgetting can purge obsolete information, reducing cognitive load and shifting the "focus of attention" of mental processes. Forgetting provides selective cognitive attention. During the process of forgetting old items, cognitive processes are effectively directed away from knowledge of minimal importance.

II.2.1.2 Explicit Focus

Humans are able to direct their attention, and potentially the sensory channels that attention relies upon, based on the surrounding environment and the dynamics of present tasks. The ability to rapidly redirect human attention is critical for level one SA and as a result has an effect on level two and three SA. When attention is intentionally placed on particular aspects within the environment, the human's knowledge of those items may increase while information regarding other items will become stale, and potentially forgotten. Selective attention can filter extraneous information from the environment, but may also prevent critical knowledge from being detected and understood. Attention must also be focused on the items critical to the success of current tasks and goals. Without focused attention, distractors prevent the consistent acquisition of information critical for the development and maintenance of SA (Smart, 2005). In the extreme; however, focused attention becomes cognitive tunneling (Wickens and Hollands, 2000), ignoring critical information to focus on one particular item in the environment. In complex, dynamic environments, numerous objects within the environment require simultaneous monitoring. The ability to divide attention between these items is paramount to the development of SA. Affording attention to unnecessary objects can degrade SA. Imperfect sampling and failed data retrievals may require repeated sampling (Wickens and Hollands, 2000).

Forgetting and directed attention are complimentary processes that provide many of the same benefits to humans operating within complex domains. A common characteristic of these domains, information overload hampers SA generation abilities, but selective attention can provide data filters minimizing this knowledge burden. Forgetting capabilities may allow for the removal of erroneous and unnecessary data passing through these attentive filters, further minimizing the quantity of unnecessary information.

II.2.1.3 Short and Long-Term Memory

Humans possess short-term and long-term memory that stores previous experiences and information required to achieve and maintain SA (Freedman and Adams, 2007). A number of models of short-term and long-term memory exist, varying between two philosophies, that short-term and long-term memory are separate and distinct systems and that both forms of memory are two aspects of the same system (Jonides et al., 2008). Baddeley (1986, 2000) developed one of the most influential multi-store models of human memory, which describes a collection of model buffers working together to form short-term memory. Interactions between short-term and long-term memory are coordinated by a central executive module (Jonides et al., 2008). Unitary-store models of short-term memory, consisting of activated long-term memory representations, may have originally been proposed by Atkinson and Shiffrin (Jonides et al., 2008; Atkinson and Shiffrin, 1971). In these models, short-term memory consists of activated long-term memory representations and recently acquired sensory percepts. Representations within long-term memory possess varying activation levels whose strength fluctuates depending on observation frequency and recency. A relationship exists between the activation strength of a long-term memory concept and that item's probability of being in short-term memory. As activation strength increases, so does the chance the item will be activated and included in short-term memory. A restricted set of the most activated short-term memory items are within the focus of attention and are directly accessible for cognitive processing (Jonides et al., 2008; Oberauer, 2006).

Short-term memory provides rapid access to information, but is capacity limited. Multi-store models of memory interpret the capacity limitations of short-term memory as the result of the rate of forgetting combined with the speed of rehearsal and discovery. Unitary-store models of memory present this limitation as a limit on the number of items that can be activated from long-term memory at any point in time (Jonides et al., 2008). Long-term memory can be unreliable, at times failing to recall information or returning incorrect data (Freedman and Adams, 2007).

Both short-term and long-term memory are critical for the development and maintenance of SA. After information has been distilled from the environment, it is placed in short-term memory and forms level one SA. If percepts are rehearsed or repeatedly perceived, their encoding strength and durability will increase, otherwise they will be forgotten (Jonides et al., 2008). Forgetting induced short-term memory instability and capacity limitations, are often viewed as negative forces working against human performance (Wickens and Hollands, 2000), but may serve critical functions in the development and maintenance of SA. Tasks and goals evolve while objects in the environment change. Forgetting provides a mechanism for removing this old information; reducing the interference and erroneous memory recalls that prevent rapid recall of contextually important information.

Procedural information residing in long-term memory, including schemata (Kumar, 1971), mental models (Minsky, 1974), semantic networks (Sanford and Garrod, 1981), and frames (Minsky, 1974), combine with level one SA to form a comprehensive understanding of the environment and predictions of the short-term evolution of that environment (level two and level three SA). While fairly stable, phenomena and facts represented in long-term memory can evolve, both rapidly and gradually. Forgetting of long-term memory elements allows adaptation to changing conditions, such as technological improvements, and minimization of the effects of false beliefs that were previously thought to be true.

II.2.1.4 Failures

Humans are prone to a number of physical and cognitive errors, mistakes, and failures. To effectively operate within real world environments, humans need to find means to effectively deal with these failures, non-optimal decision making, and unintentional events. Failures can result in reduced SA and often are correlated with inaccurate or insufficient SA (Freedman and Adams, 2007). Even the best, highly trained, professional athletes, humans who have dedicated potentially their entire lives to being able to perform a very small set of tasks, cannot remove all sources of uncertainty (Davids et al., 1991). Mental ability also suffers the curse of failure. Forgetting provides a mechanism for erroneous decisions and inaccurate mental constructs to be removed from memory; increasing the ability to recall vital and correct information.

II.2.1.5 Goals, Plans, and Their Evolution

In complex dynamic environments, it is quite common to simultaneously maintain multiple goals, which may or may not conflict at some level. These goals not only evolve over time, but require prioritization based on their importance to the overall system task. When utilizing a data driven processing method, patterns may emerge from the environment, dictating whether new goals must be created, existing goals should be changed, or if priorities require alteration. Conversely, a goal driven methodology will influence how new observations from the environment are integrated and interpreted.

Goals can be thought of as ideal system states, which are used to select the most appropriate mental model for a given task. The resultant combination of goals and mental models allows for plans to be generated that modify the environment to align with key environmental states. If available, scripts may also be used to assist in the action selection process.

Current goals may be malleable, evolving as a result of performance and interactions with the environment. When the environment and current mental models are consistent, correct operation is commonly assumed, but when differences exist, plans and goals may require modification. Additionally, the relative importance of goals and plans may shift in an attempt to better align the environment and mental models. Figure II.4 graphically describes the relationship between goals, plans, mental models, and SA (Endsley, 1995b).

As goals change, knowledge requirements can similarly transform. Forgetting can provide a means of removing information that is no longer relevant to current goals and increase the ability to recall new germane observations from the environment. When switching tasks, humans frequently experience a temporary performance penalty during the initial portion of new tasks, often labeled “switch-cost” or “restart cost” (Allport and Wylie, 2000). Altmann described the theory of functional decay, which describes how within-run slowing can minimize “switch-cost”. Within-run slowing appears as a progressive decrement in performance during task execution. This reduction in performance results from memory items critical to task completion becoming progressively more challenging to recall. While task performance declines on the current task, the acquisition and retention of knowledge required for a subsequent task is improved (Altmann, 2002; Altmann and Gray, 2002; Altmann and Schunn, 2002; Sims and Gray, 2004).

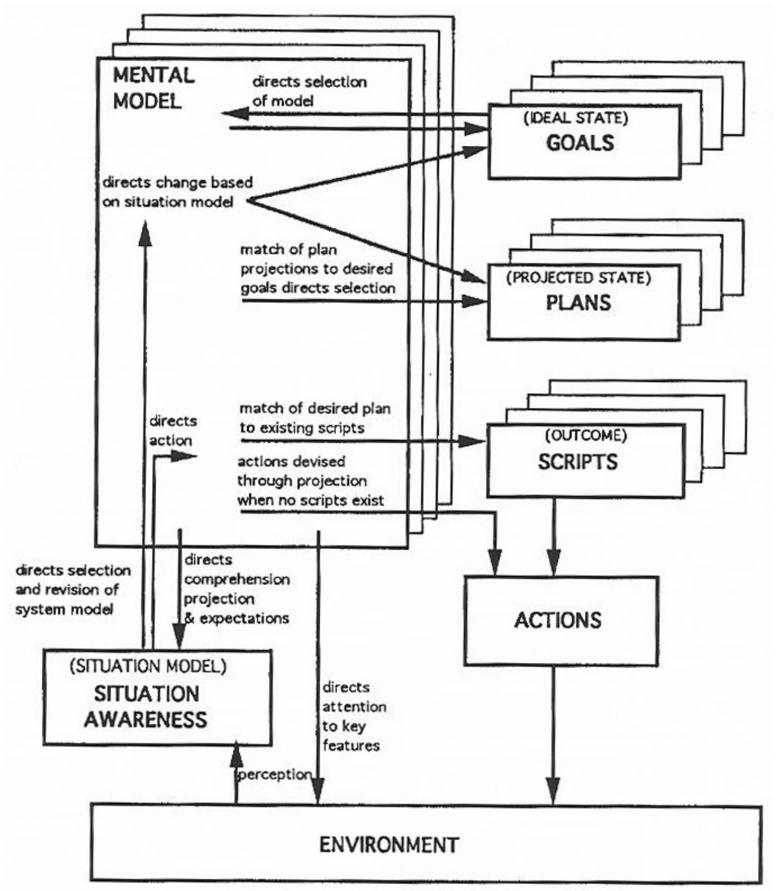

MENTAL MODEL
directs selection of model
(IDEAL STATE) GOALS
directs change based on situation model
match of plan projections to desired goals directs selection
(PROJECTED STATE) PLANS
directs action
match of desired plan to existing scripts
(OUTCOME) SCRIPTS
actions devised through projection when no scripts exist
directs selection and revision of system model
directs comprehension projection & expectations
ACTIONS
(SITUATION MODEL) SITUATION AWARENESS
directs attention to key features
perception
ENVIRONMENT

Figure II.4: Relationship of goals, plans, and mental models to SA (Endsley, 1995b)

SA is a complex phenomena comprised of many cognitive and psychological components. While not a process in and of itself, SA provides a grouping of mental processes enabling humans to operate within complex and dynamic domains. The majority of SA components are affected by data management issues and forgetting may provide a means of removing unnecessary and obsolete information. With less data to process, humans are better equipped to achieve high levels of SA and performance. This section presented several components of SA and how they benefit from forgetting. Appendix B provides descriptions of additional SA components.

II.2.2 Robotic Situational Awareness

Robots are often tasked to operate in the same complex and dynamic domains as humans. SA can assist robotic systems in many of the ways that humans benefit from the psychological phenomenon. A large body of work exists that aims to improve the SA of human remote operators that work with robots, but research aspiring to achieve human levels of SA within robots is difficult to find.

II.2.2.1 Improving Remote Operator Situational Awareness

Considerable effort in the field of Human-Robot Interaction (HRI) has focused on improving the SA of humans interacting with robots (Drury et al., 2003; Humphrey and Adams, 2009a; Kaber et al., 2000; Burke and Murphy, 2004; Scholtz et al., 2005; Sellner et al., 2006; Stubbs et al., 2007; Riley and Endsley, 2005; Gatsoulis, 2008). In some cases, remote operators directly tele-operate robots to complete tasks and acquire sensor readings. At other times, humans and robots operate in more of a team fashion, with a human generally assuming the leadership role.

Across this spectrum of configurations, robots can operate as various levels of autonomy. A number of taxonomies have been created to identify individual levels of autonomy (Sheridan and Verplank, 1978; Kaber et al., 2000; Parasuraman et al., 2000). Sheridan and Verplank (1978) developed the ten levels of autonomy available to a human working with a system. Shown in Table II.2, the taxonomy ranges from humans assuming complete control at one extreme, to systems possessing full responsibility at the other. Research has shown; however, that increasing the level of automation in a system may generate mixed results. In some domains and tasks, operator SA may improve but in others, unintended consequences may result, including a net loss of SA (Parasuraman et al., 2000; Adams, 2007). A primary cause of lost SA, “out of the loop” effects may appear if the level of automation is improperly increased. In this condition, the operator loses awareness of some of the events occurring within the environment as the automated system performs action selection and execution.

A number of taxonomies for multi-robot systems have been developed. Gerkey and Matarić (2003),

Table II.2: Levels of autonomy (Sheridan and Verplank, 1978)

Level	Description
10	The system is fully autonomous and ignores human input.
9	The system can optionally inform the human.
8	The system provides information by request.
7	The system can perform any action but then must inform the human.
6	The system can perform any action unless the human rejects the action selection.
5	The system requires approval from the human.
4	The system provides a single suggested action.
3	The system provides a small set of options to the human.
2	The system provides a complete list of possible actions.
1	The system does not provide assistance.

Table II.3: Types of human-robot teams (Yanco and Drury, 2004)

Type	Human	Robot
1	Human	Robot
2	Human	Robot Team
3	Human	Robots
4	Human Team	Robot
5	Humans	Robot
6	Human Team	Robot Team
7	Human Team	Robots
8	Humans	Robot Team

Dudek et al. (2002), and Balch (2002) have generated taxonomies for robot only teams that highlight the differences in communication and information requirements between differing configurations of robots. Yanco and Drury (2004) expanded these ideas to a taxonomy specifically for human-robot teams, see Table II.3. Drury et al. (2003) analyzed the SA requirements and properties of real-time HRI teams. They found that HRI tasks have a non-symmetrical property, the requirements for human teammates were not identical to the robot teammate requirements. They developed a general definition of HRI SA and a five element classification system to define the SA requirements of HRI teams, see Table II.4. The applicability of these HRI definitions for potential future systems, where robots may assume a greater share of leadership responsibilities is uncertain. In future systems, a greater emphasis on protecting humans from harm will exist, but the asymmetrical nature of HRI applications may diminish, requiring these definitions to be modified.

A large body of work exists exploring how communication interfaces can be developed to increase the level of human operator SA (Adams, 2009; Goodrich et al., 2008; Adams and Freedman, 2007; Freedman and Adams, 2007; Gatsoulis, 2008; Sellner et al., 2006; Murphy et al., 2008; Scholtz et al., 2004; Yanco and Drury, 2004; Scholtz et al., 2005; Humphrey and Adams, 2009a). Under conditions where operator SA is

Table II.4: Definitions of HRI SA forms (Drury et al., 2003)

Name	Definition
HRI Awareness (General Case)	"Given *n* humans and *m* robots working together on a synchronous task, HRI awareness consists of five components:"
Human-Robot Awareness	"The understanding that the humans have of the locations, identities, activities, status, and surroundings of the robots. Further, the understanding of the certainty with which humans know the aforementioned information."
Human-Human Awareness	"The understanding that the humans have of the locations, identities, and activities of their fellow human collaborators."
Robot-Human Awareness	"The robots' knowledge of the humans' commands needed to direct activities and any human-delineated constraints that may require command noncompliance or a modified course of action."
Robot-Robot Awareness	"The knowledge that the robots have of the commands given to them, if any, by other robots, the tactical plans of the other robots, and the robot-to-robot coordination necessary to dynamically reallocate tasks among robots if necessary."
Humans' Overall Mission Awareness	"The humans' understanding of the overall goals of the joint human-robot activities and the measurement of the moment-by-moment progress obtained against the goals."

commonly explored, two primary forms of HRI exist, remote interaction and proximate interaction (Goodrich and Schultz, 2007). Under remote conditions, operators are not co-located with the robots and all interaction occurs through a computer interface. Examples for modifying a graphical user interface to support SA during remote interaction include the exploration into novel compass visualizations (Humphrey and Adams, 2008) and the creation of guidelines governing interface design (Scholtz et al., 2004). Research aiming to improve human operator SA during proximate interactions, when the human and robot(s) are co-located, includes the creation of specific roles for human operators to assume (Murphy et al., 2008; Scholtz et al., 2005; Goodrich and Schultz, 2007) and providing robots with anticipatory capabilities to improve team efficiency (Hoffman and Breazeal, 2007).

II.2.2.2 Improving Robotic Situational Awareness

While considerable work and progress has been made towards improving human SA when operating within human-robot teams, a far smaller body of work exists directly attempting to increase the level of SA possessed by robots. To some extent; however, a large portion of robotics research has indirectly worked towards the development of human level SA. Improved hardware-based sensors allow for increased accuracy of perception, including the minimization of missed data located within the environment. Mapping and navigation algorithms, such as Simultaneous Localization And Mapping (Smith et al., 1990), FastSLAM (Thrun et al., 2004), and Markov localization (Fox et al., 1999), allow robots to operate within complex, dynamic environments, sometimes with no a priori information. Using these algorithms, robots can maintain a sense of localization and simultaneously develop a comprehensive map of their surroundings. Research has been conducted to allow robots to gauge and estimate the psychological and emotional state of humans located within close proximity (Mower et al., 2007; Breazeal et al., 2009). Even research not inherently associated with robotics, such as machine vision (Davies, 2004) and natural language processing (Manning and Schütze, 1999), has increased the ability for robots to develop SA. This body of work is gradually advancing the SA capabilities of robots from a bottom-up perspective.

The body of robotics research aiming to explicitly achieve human levels of robot SA from top-down or holistic approaches is scarce. Gatsoulis and Virk (2006) suggested taking a holistic approach to improving performance within the Urban Search and Rescue domain by improving the SA of both humans and robots. By increasing the level of SA within human operators and within the controlled robots, the SA of the entire system may be improved, resulting in increased performance. In their presented work, only efforts to understand, measure, and improve operator SA were presented. The desire to improve the SA of robots operating within the Urban Search and Rescue domain has been reiterated in the future work section of Gatsoulis's PhD thesis (Gatsoulis, 2008).

Lison et al. (2010) recently crafted the ability for robots employing the CoSy Cognitive Architecture, see Section II.3.2 for a full review, to create and maintain rich probabilistic multi-modal belief models from low-level sensory information. These hierarchical structures are founded on Markov Logic (Richardson and Domingos, 2006) and allow for comprehensive situated reasoning.

II.3 Cognitive Architectures

Contemporary designs for intelligent robots often utilize cognitive architectures. Considerable work has been completed towards the realization of cognitive agents (McCarthy, 1959; Cox, 2005; McCarthy et al., 2002; Cox, 2007), but current systems have not come close to achieving human-level Artificial Intelligence (AI) (McCarthy et al., 2002). The vast majority of current AI systems use only a limited number of algorithms and representations, normally only one (McCarthy et al., 2002), but a symbiotic relationship of multiple, and possibly many, different representations and algorithmic approaches may need to be formed (Minsky, 1992). The Causality Matrix, see Figure II.5, is an initial attempt to categorize classes of algorithms and determine how they can be combined (Minsky, 1992). To achieve human-level AI, different algorithms and representations will need to be assimilated into one coherent and diverse system (McCarthy et al., 2002; Cassimatis et al., 2006), plus the system must be capable of seamlessly transitioning between algorithmic processes while still enabling parallel processing (Singh and Minsky, 2003).

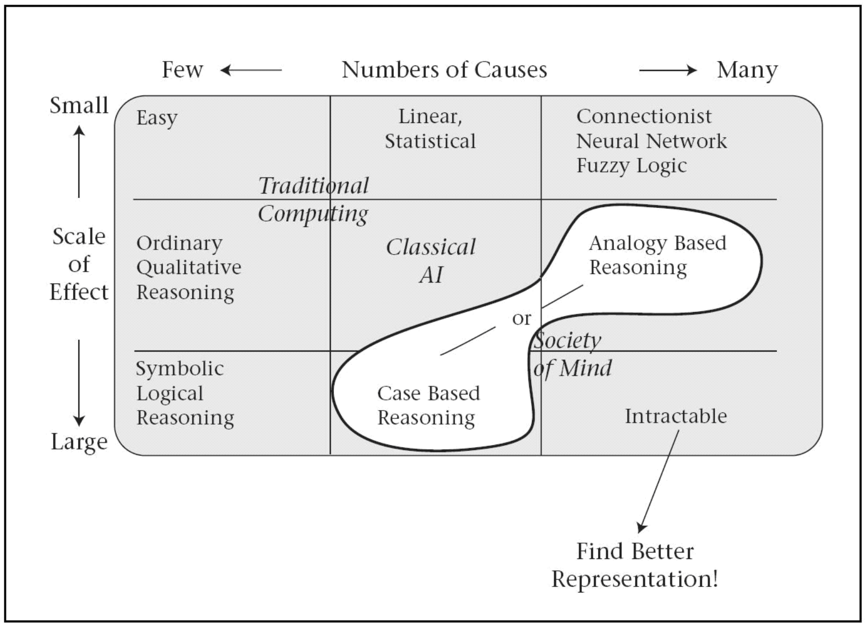


Figure II.5: The Causality Matrix (Minsky et al., 2004)

Robots capable of generating human levels of SA within complex, dynamic domains will require diverse cognitive and physical capabilities. Forgetting may increase robotic capabilities by improving the effectiveness and efficiency of many of the cognitive processes required for robotic SA generation. Cognitive architectures provide principled means for enabling forgetting to aid these processes. This section presents a number of existing cognitive architectures designed for domains ranging from mobile robotics to human performance modeling. Many of the presented architectures incorporate forgetting mechanisms or have previously been incorporated into robotic systems. Several presented cognitive architectures do not inherently incorporate forgetting mechanisms, but allow for their introduction. Cognitive architectures along with forgetting mechanisms may allow robots to complete complex mental operations, minimize unnecessary data, and increase SA.

II.3.1 The Minsky-Sloman Architecture

Two potential architectures that can allow agents to easily switch between "ways to think" (Minsky, 2006) are the Human-CognitionAffect (H-CogAff) architecture (Sloman, 2001, 2003), see Figure II.6, and the Model Six architecture (Minsky, 2006; Singh, 2005), see Figure II.7. These frameworks share many components in common and are sometimes collectively referred to as the Minsky-Sloman model (Minsky et al., 2004). These hierarchical architectures are flexible in that the number, type, and format of cognitive processes are allowed

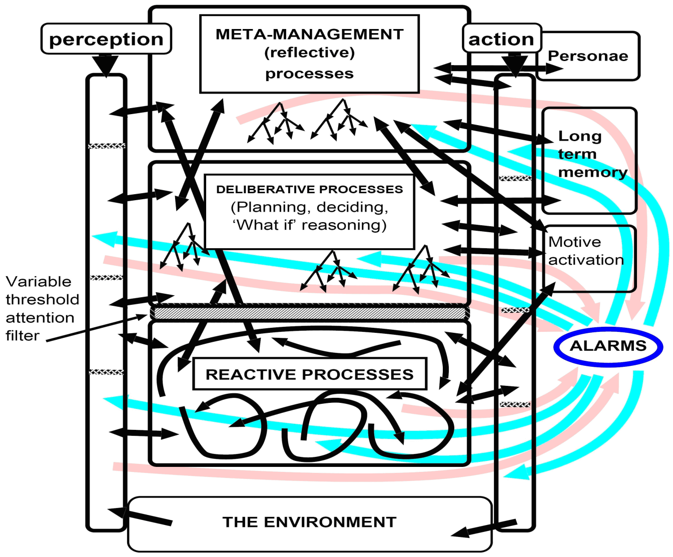

perception
META-MANAGEMENT (reflective) processes
action
Personae
Long term memory
DELIBERATIVE PROCESSES (Planning, deciding, 'What if' reasoning)
Motive activation
Variable threshold attention filter
ALARMS
REACTIVE PROCESSES
THE ENVIRONMENT

Figure II.6: The H-CogAff Architecture (Sloman, 2001)

to vary across implementations. Processes can be executed in parallel and hierarchical level boundaries are not absolute. The reactive layer represents processes that immediately form responses to sensed values. Deliberative processes examine multiple responses to input and determine the most appropriate action. The meta-management layer of the H-CogAff architecture represents similar processes to the top three levels of the Model Six architecture. Meta-management processes provide various forms of metacognition through controlling, monitoring, and evaluating self (McCarthy et al., 2002). Forgetting mechanisms are not inherent to these architectures, but can be incorporated into individual instances.

The EM-ONE architecture for reflective commonsense reasoning provides an initial implementation of the Model Six architecture (Singh, 2005). Realizing the lower three levels of the Model Six architecture, EM-ONE provides commonsense reasoning capabilities and the benefits of Minsky's critic-selector model (Minsky, 2006). EM-ONE incorporates ideas from the H-CogAff architecture and has been used to control two simulated robots working cooperatively on physical manipulation and social interaction tasks (Singh, 2005).

II.3.2 CoSy Architecture Schema

The Cognitive Systems for Cognitive Assistants (CoSy) Architecture Schema, is a modular schema designed to allow for the creation of agents capable of parallel modular processing, metacognition, and structured data management. Each loosely coupled module, see Figure II.8, is referred to as a subarchitecture and generally works within one realm of reasoning, visual, spatial, etc. Subarchitectures consist of four separate types of components. Unmanaged processes are generally light weight and run continuously. Working memory stores

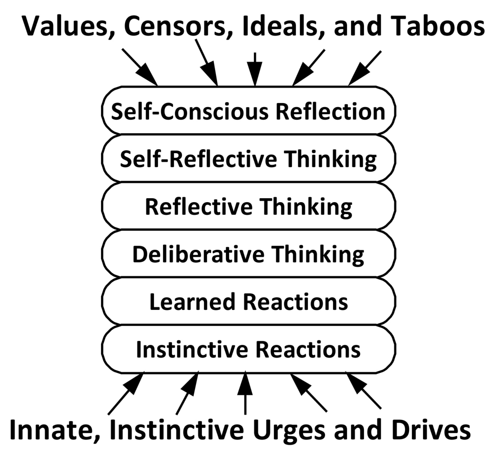


Figure II.7: The Model Six Architecture (Recreated from (McCarthy et al., 2002))

data derived from the unmanaged components and is writable by any component within the module, but read-only to the rest of the system. However, some privileged components may exist that can modify other modules' working memory. Managed processes, which generally perform expensive calculations, monitor working memory for relevant changes. When these changes are detected, the managed processes determine what actions should be completed. Permission is required from the task manager before a managed component can begin any new processing. Cognitive Systems for Cognitive Assistants Architecture Schema agents require a subarchitecture designed to perform coordination and another to bind symbols. Binding enables high-level inter-module communication. The schema does not inherently incorporate many of the capabilities present in other cognitive architectures that directly model human cognition (such as forgetting), but it does allow for their inclusion through the development of appropriate subarchitectures (Hawes et al., 2006, 2007). This schema has been incorporated into robotic systems capable of human-robot interaction, completing physical manipulation tasks, and exploratory behavior (Hawes et al., 2010, 2009, 2007; Jacobsson et al., 2008; Kruijff et al., 2007). The ability for these robots to generate and maintain rich belief models has recently been crafted (Lison et al., 2010). These models are spatio-temporally framed structures incorporating epistemic information and utilize Markov Logic (Richardson and Domingos, 2006). Similar to systems designed for commonsense reasoning (Mueller, 2006), this addition allows for reasoning over situationally dependent information.

II.3.3 Polyscheme

Polyscheme (Cassimatis, 2002; Cassimatis et al., 2007) is a flexible hybrid cognitive architecture designed to tightly integrate diverse arrays of knowledge representations and inference schemes. Five central principles,

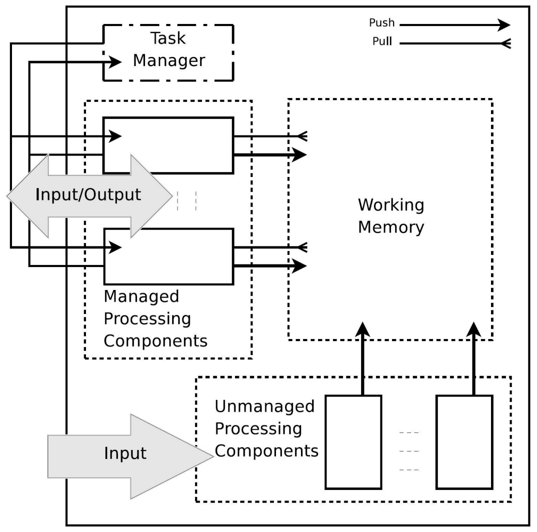

Push
Pull
Task Manager
Input/Output
Working Memory
Managed Processing Components
Unmanaged Processing Components
Input

Figure II.8: The CoSy Architecture Schema Subarchitecture (Sridharan et al., 2007)

1. Algorithms and representations have niches
2. Frequent communication between components is critical
3. Focus of attention must be carefully guided and singular
4. The Common Function Principle
5. The Multiple Implementation Principle

Figure II.9: Guiding principles of Polyscheme (Cassimatis, 2002)

shown in Figure II.9, have guided the development of Polyscheme (Cassimatis, 2002, 2006). Polyscheme is composed of three components, a set of specialists, an attention buffer, and a focus manager. Specialists implement inference schemes, and except for supporting a communications protocol, their design is unrestricted. Specialists reason over and communicate with knowledge chunks called propositions. The attention buffer stores a queue of these propositions, which have been requested for focus by specialists. The focus manager selects propositions and forces each specialist to perform inferences on them (Cassimatis et al., 2007). With this design, specialists are able to utilize different algorithms and representations, but stay abreast of knowledge changes within the system. Polyscheme's goal is to form a commonsense substrate, a means of simplifying the realization of complex AI problems by providing a common base architecture capable of supporting a diverse array of domains (Cassimatis, 2002). A review of available literature did not reveal any experiments directly relating Polyscheme to beneficial forgetting capabilities.

Polyscheme provides a systematic means for integrating different forms of inference and knowledge representations. Through focusing, knowledge can be shared across data representations. Polyscheme's focus manager is domain independent and different attention fixation strategies can be utilized to modify and control the operations of specialists (Cassimatis et al., 2007), making the architecture flexible and applicable to numerous domains requiring complex cognitive ability. In a demonstration of the ease of adapting Polyscheme to different domains, the original implementation, which was concerned with physical reasoning, was converted into a natural language syntactic parser (Cassimatis, 2006). Polyscheme has also been used in human-robot interaction and heterogeneous information retrieval applications. Real robot hardware experiments have been completed using Polyscheme (Cassimatis et al., 2007), including NASA's Robonaut humanoid robot (Sofge et al., 2004; Ambrose et al., 2000).

II.3.4 ICARUS

ICARUS (Choi et al., 2004; Langley and Choi, 2006) is a cognitive architecture aiming to realize qualitative aspects of human behavior. Several concepts regarding human behavior that originated in the field of psychology have influenced the design of ICARUS. These include, the use of means-ends analysis to solve unfamiliar problems, mental problem solving combined with execution to complete challenges within a physical context, skill generation resulting from the solving of novel challenges, and converting backward-chaining search into a forward-chaining process through learning (Langley and Choi, 2006). ICARUS utilizes two forms of knowledge, concepts and skills. Concepts represent forms of environmental situations built from lower level concepts and percepts. Skills describe how goals can be decomposed into subgoals. Long term memory possesses a hierarchical structure of skills. Concepts utilize relations between perceptions, while skills are composed of relations between actions. Skills incorporate goal conditions, requisite preconditions, and continuation conditions (Langley et al., 2009). Recently, episodic memory capabilities have been added to ICARUS (Stracuzzi et al., 2009).

ICARUS follows a recognize-act paradigm. First, a perceptual buffer is populated with percepts of objects perceived within the environment (Langley et al., 2009). These environmental percepts include a type, name, and attribute pairs that describe the object in the environment being represented by the percept (Langley and Choi, 2006). Low-level concepts are then compared against these percepts and matches are transfered to short-term memory in the form of beliefs. Higher level concepts may be relevant to the newly created beliefs, resulting in a cyclical belief creation pattern (Langley et al., 2009) Categories within ICARUS are represented in a boolean fashion, preventing fuzzy classification (Langley and Choi, 2006). Starting from the agent's top-level goal, a search through its skill hierarchy is taken with paths containing skills possessing fulfilled preconditions, but unsatisfied goals. Once a primitive skill is reached that possesses actions available for direct execution, the search is terminated and the actions are taken. When the search fails to find primitive actions to execute, the agent begins problem solving, which uses backward chaining to achieve goals and generate new skills (Langley et al., 2009).

ICARUS has been used in domains requiring inference, action execution, problem solving, and learning. Some of these domains include logic problems (Tower of Hanoi, Multi-column subtraction, FreeCell, and logistics), control of synthetic characters (Langley et al., 2009), and in-city driving (Langley and Choi, 2006). Currently, the feasibility of using ICARUS to control robots is being explored (Langley et al., 2009).

II.3.5 ACT-R

Adaptive Control of Thought - Rational (ACT-R) (Anderson et al., 2004; Anderson and Lebiere, 1998) is a cognitive architecture designed to model human behavior. The architecture is composed of a collection of modules, each representing a different type of knowledge and functionality. These modules include sensor processing modules, motor modules, an intentional module representing system goals, and a declarative module that stores long-term declarative knowledge. Each module has a buffer that acts like short-term memory, holding a single unit of relational declarative knowledge, frequently referred to as a chunk. Long-term memory in ACT-R consists of a collection of production rules used to coordinate module processing. When long-term memory productions match chunks in short-term memory, module buffers may be modified, new structures may be constructed in memory, and actions within modules may be taken (Anderson et al., 2004).

Production matching within ACT-R is founded on an activation level approach. Chunks in memory compete for selection by productions based on an activation level set by equations II.1 and II.2. These equations calculate the activation of a chunk based on a base activation (B_i) and the chunk's relevance to the current context. W_j, source activation, represents attention directed to portions of the current goal and S_{ji}, strength of association, represents the frequency that a chunk was required when element j was part of a current goal. ε_1 and ε_2 represent permanent and transient noise, respectively. The base level activation (B_i) is calculated by combining an initial expected base level activation value (β) with a logarithmic function based on the retrieval times of a chunk (Anderson and Lebiere, 1998).

$$A_i = B_i + \sum_j (W_j S_{ji}) + \varepsilon_1 + \varepsilon_2 \tag{II.1}$$

$$B_i = \beta + ln(\sum_{k=1}^{n} t_k^{-d}) \tag{II.2}$$

$$P_i = \frac{1}{1 + e^{-(A_i - \tau)/s}} \tag{II.3}$$

$$T_i = Fe^{-A_i} \tag{II.4}$$

Chunk selection is not a deterministic process. After chunk activation levels have been updated, two equations are used to select a chunk for activation and determine the latency before the chunk becomes activated. Equation II.3 calculates the probability that a chunk's activation level will exceed a preset threshold value (τ). τ acts as a minimum threshold value for an item's activation level, while s provides noise to the system. When the activation of multiple chunks exceeds τ, the final chunk is selected probabilistically. A retrieval latency, defined by equation II.4, delays the activation of the selected chunk. F provides a scaling factor to match units to real time (Taatgen et al., 2006).

ACT-R inherently incorporates the concept of forgetting. Equation II.2, which calculates a chunk's current base activation level, implements trace-decay (time-based forgetting) and derivatives of the function have been used to model trace-decay by other architectures and models (Nuxoll et al., 2004; Schooler and Hertwig, 2005). The presence of the summation term in equation II.1 results in the ACT-R model possessing similarity-based interference (see Section II.4 for a description of the different types of interference that may lead to forgetting). Similarity-based interference, interference resulting from the presence of items with similar properties, results from the strength of association terms, $S_{i,j}$, whose values are dependent on the number of chunks associated with components of the current goal and the total number of chunks in declarative memory. As the number of similar chunks increases, the strength of association decreases, resulting in increased interference. The combination of base level activations and encoding interference enables ACT-R to directly model the psychological notion of "fan-out" (Anderson and Lebiere, 1998).

ACT-R is capable of both structural and statistical learning. Base activation levels for chunks evolve through the operation of the ACT-R system. Frequently selecting a chunk will increase its base level activation, while ignoring a chunk will cause its activation level to fade. Through a process of production compilation, new productions can be generated. New productions are generated based on the firing of existing productions and variable replacement (Langley et al., 2009).

ACT-R has been used across a wide variety of domains within experimental psychology. Phenomena that have been explored using ACT-R include memory, attention, reasoning, problem solving, and language processing (Langley et al., 2009). ACT-R has also been used to explore if forgetting is beneficial to heuristics (Schooler and Hertwig, 2005) and to model individual differences (Daily et al., 2001). A number of robotic applications have been developed using ACT-R. Burghart et al. (2006) incorporated ACT-R into a robotic system that solves jigsaw puzzles with the assistance of a human tutor. ACT-R/S (Harrison and

Schunn, 2003; Harrison, 2007), an extended version of ACT-R that incorporates two spatial buffers (configural and manipulative), has been incorporated into NASA's Robonaut project (Sofge et al., 2004). Another extension of ACT-R, ACT-R/Embodied (ACT-R/E) (Trafton et al., 2008), combines visual and auditory data to enable robots to track conversations within human-robot interaction domains.

II.3.6 Soar

Soar (Lehman et al., 2006)[1]is a cognitive architecture originating in the 1980's that aims to model high-level aspects of human cognition (Laird, 2008). Two fundamental assumptions have guided Soar's development, that humans are complex knowledge systems that rationally utilize knowledge, and that human cognition primarily operates at the symbolic level (Johnson, 1997). Soar possesses three forms of long-term memory: procedural, semantic, and episodic knowledge (Nuxoll and Laird, 2007; Laird, 2008). Procedural memory is composed of operators that describe primitive and high level domain specific actions. Operators can propose actions both internal and external to an agent. Working memory within Soar contains declarative knowledge and is the only form of memory that can be matched by procedural knowledge (Johnson, 1997).

Tasks in Soar are formed around the achievement of goals. Goals are stored on a stack and can possess subgoals to form a hierarchical structure. The processing cycle within Soar consists of three steps, propose operators, select an operator, and apply the selected operator. During the process of selecting operators that have been suggested by procedural memory, if multiple operators have the same preference or if no operators have been proposed, then an impasse is instigated (Langley et al., 2009). During an impasse, a new goal aiming to solve the impasse is created and is added to the agent's goal stack (Johnson, 1997). When a new goal is added to the stack, a new state is generated, which inherits the properties of the parent state (Langley et al., 2009). Operators are able to modify parent states, both directly and indirectly through external actions. When a goal is achieved, the goal is removed from the stack and the substate is discarded (Laird, 2006).

The original design for Soar did not incorporate forgetting, but it has subsequently been added (Chong, 2003). Chong's implementation of forgetting borrowed the activation equation from ACT-R, see equations II.1 and II.2. The spreading activation term in equation II.1 and the permanent noise term were not used. The summation term was not included in this implementation because Soar did not possess any sub-symbolic associative links. Equation II.5 defines the value for the transient noise term, where *ans* represents the variance of the distribution used to model the noise within the system. The decay implementation in Soar was then extend by Nuxoll et al. (2004). Extensions to Chong's implementation included improved efficiency of the decay features, extending decay functionality to the entirety of working memory, and applying acti-

[1]Soar originally stood for State, Operator, And Result, but the name is no longer considered to be an acronym. Additionally, Soar is no longer written in all capital letters to reflect Soar's transition to a proper name (Ritter and Kim, 2006).

vation differently to both forms of working memory present within Soar. Additionally, a change was made that diverges from the original ACT-R implementation. Instead of setting the initial base activation of a new working memory item to a fixed value, the item's initial activation level is dependent on the activation levels of the working memory elements that caused the item to be created.

$$\begin{aligned} \varepsilon_2 &= ans * \log\left[(1.0 - p)/p\right] \\ p &= \text{rand}\left[0.0, 1.0\right] \end{aligned} \tag{II.5}$$

Soar is capable of procedural, episodic, and semantic learning. Procedural learning occurs via two processes, chunking and reinforcement learning. When subgoals are completed, chunking can be used to abstract the processing that resulted in the goal being completed. This abstraction is then converted into a new production rule. Reinforcement learning adjusts the operator selection process by modifying operator preference levels. Semantic memory is bolstered through the accumulation of portions of working memory for later use, while episodic learning records copies of working memory at different points in time (Langley et al., 2009).

A diverse array of domains have been explored with the Soar architecture (Langley et al., 2009). Software agents using Soar have been developed to compete in the First Person Shooter style computer games Quake 2 (Laird, 2001), Unreal Tournament (Magerko et al., 2004), Quake 3, and Descent 3 (Wintermute et al., 2007). Soar has also been used to model military pilots participating in air combat training exercises (Tambe et al., 1995). Human language processing (Lewis, 1993) and categorization (Miller and Laird, 1996) have also been explored with the Soar cognitive architecture. Soar has previously been employed by robotic systems. Sultanik et al. (2008) developed a system where Soar controlled mobile robots performing team-based surveillance. These robots interacted with human operators through a Personal Digital Assistant (PDA) based interface. Hanford et al. (2008) developed a six-legged hexapod robot that used Soar to complete navigation and obstacle avoidance tasks.

II.3.7 CLARION

Connectionist Learning with Adaptive Rule Induction ON-line (CLARION) (Sun, 2003) is a fixed-hybrid cognitive architecture composed of four distinct subsystems, the Action-Centered Subsystem, the Non-Action-Centered Subsystem, the Motivational Subsystem, and the Metacognitive Subsystem, see Figure II.10. The Action-Centered Subsystem performs both physical and mental action control, while the Non-Action-Centered Subsystem maintains non-action centric knowledge. Motivational decision support is provided by the Motivational Subsystem. The Metacognitive Subsystem monitors, directs, and alters the actions of the other subsystems. All subsystems provide both implicit and explicit forms of knowledge representation. Implicit knowledge resides in the "bottom" level of each subsystem and consists of sub-symbolic reasoning

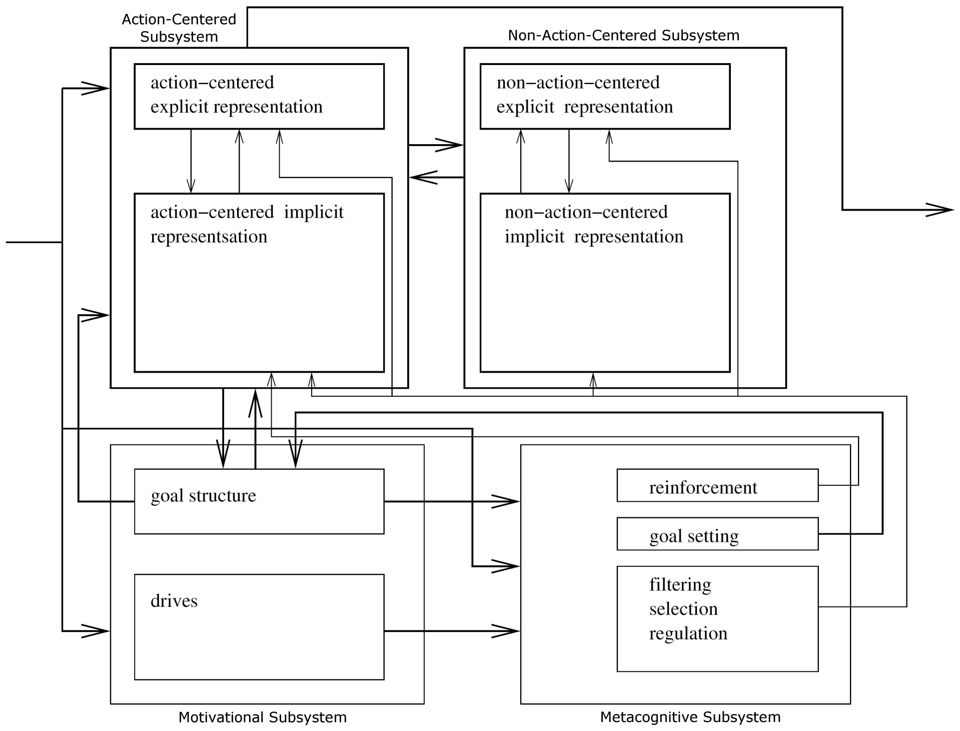

Action-Centered Subsystem
Non-Action-Centered Subsystem
action–centered explicit representation
non–action–centered explicit representation
action–centered implicit representsation
non–action–centered implicit representation
goal structure
drives
reinforcement
goal setting
filtering selection regulation
Motivational Subsystem
Metacognitive Subsystem

Figure II.10: The CLARION Cognitive Architecture (Adapted from (Sun et al., 2006))

utilizing neural networks. “Top” level reasoning provides symbolic processing through rule-based components (Sun et al., 2006). All four subsystems can perform bidirectional learning; top-down and bottom-up (Sun, 2006).

CLARION directly incorporates forgetting mechanisms into the Non-Action-Centered and Action-Centered subsystems. Symbolic knowledge within both subsystems is composed of a collection of chunks and associated rules, each of which can undergo forgetting. Within the Non-Action-Centered Subsystem, forgetting occurs by two separate processes. The first involves a time dependent base level activation (i.e., decay) similar to ACT-R, see Section II.3.5. Defined by equation II.6, base level activation within the Non-Action-Centered Subsystem affects the selection of chunks and associative rules. If a base level activation drops below a threshold value, the item is filtered. Rules and chunks can also be forgotten using a frequency threshold. CLARION calculates the frequency that chunks and rules are invocated (encoding, re-encoding, extraction, re-extraction, and activation) and if the invocation frequency for an item drops below some threshold, the item is removed. When chunks are removed from the system, associative rules connected to the chunk are also removed. The Action-Centered Subsystem utilizes the same forgetting mechanisms except that items within the Action-Centered Subsystem’s capacity-limited working memory can also have chunks deliberatively added and removed (Sun, 2003).

$$B_j = iB_j + c\sum_{l=1}^{n} t_l^{-d} \tag{II.6}$$

CLARION explicitly provides metacognitive capabilities through its subsystems. While one subsection has been labeled the metacognition subsection, metacognitive capabilities are also provided via the motivational subsystem. The action-centered subsystem can also assist in realizing metacognition, but does not have direct architectural support (Sun, 2003; Sun et al., 2006).

CLARION's metacognitive subsystems utilize a diverse array of status information in determining how to modify the system. All modules within the metacognitive subsystem have access to the system's state, goal structure, and state of the systems drives. Through the goal setting module, the CLARION architecture is capable of generating new goals that should be pursued. The evaluation module processes the above inputs into a reinforcement reward to be used for learning. Filtering, selection, and regulation are also possible within the metacognitive subsystem. Through its reasoning, the metacognitive subsystem is able to modify many of the system's properties. Reasoning and data processing provided by CLARION's motivational subsystem also promotes metacognition. Via generating drive strengths for an agent, this subsystem indirectly determines the importance of goals at any point in time.

CLARION has previously been used to simulate a number of cognitive tasks, including serial reaction time, artificial grammar learning, process control, alphabetical arithmetic, Tower of Hanoi (Sun, 2002), minefield navigation (Sun and Peterson, 1998; Sun et al., 2001), and social simulation (Naveh and Sun, 2006). The cognitive architecture has been used to explore the interactions between rule-based reasoning, similarity-based reasoning, and associative memory (Sun et al., 2006) and also investigate the phenomenon of performance degradation under pressure (Wilson et al., 2009). Additionally, CLARION models have been employed to probe the relationship between the dichotomies of implicit versus explicit knowledge and procedural versus declarative knowledge (Sun et al., 2009). A review of the available literature did not reveal any experiments involving real robots using the CLARION architecture.

II.3.8 COGNET

COGnition as a NEtwork of Tasks (COGNET) (Harper and Zacharias, 2004; Zachary and Mentec, 2000; Zachary et al., 2005) is a real-time multi-tasking cognitive architecture designed to produce agents for intelligent training, decision-support, and human performance modeling. As shown in Figure II.11, COGNET consists of four main components: cognition, perception, and action subsystems along with a memory store. All three subsystems operate in parallel and the cognitive and perceptual subsystems can access the memory

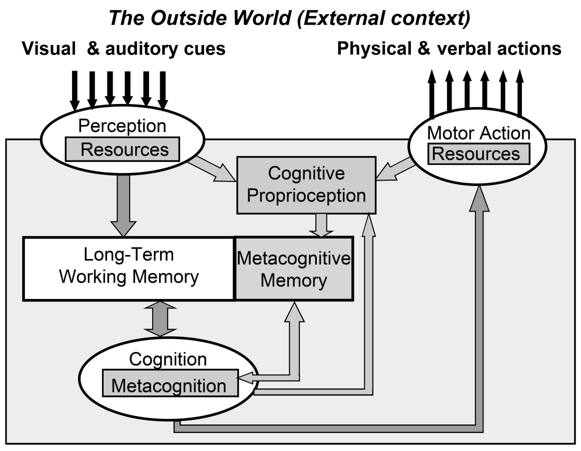


Figure II.11: The COGNET Cognitive Architecture (Zachary et al., 2005)

store independently. Sets of resources exist within the perception and motor subsystems to afford timing and accuracy simulations with the outside world. Metacognition support has been incorporated into the original design, adding a cognitive proprioception component, additions to the memory store, and metacognitive reasoning within the cognitive subsystem. For purposes of simplifying models within the COGNET architecture, the memory store operates as a long term memory structure (Zachary et al., 2005).

COGNET maintains five separate forms of knowledge: declarative, procedural, action, perceptual, and metacognitive (Zachary et al., 2005). Declarative knowledge utilizes a blackboard representation that can be modified by both the cognitive and perceptual subsystems (Zachary et al., 1998). Guiding the operations of the cognitive subsystem, modifying declarative memory, and selecting actions for the motor subsystem (Zachary et al., 2005), procedural knowledge utilizes a chunking mechanism based on Goals, Operators, Methods, and Selection Rules (GOMS) (Zachary et al., 1998). Action knowledge instructs the motor subsystem how to interact with the outside world. Rule-based perceptual knowledge consists of the form, IF event Then POST updated data to memory store (Zachary et al., 1998). Metacognitive knowledge is utilized by the cognitive subsystem and is GOMS based (Zachary et al., 2005). Metacognitive memory stores information relevant to the system, which simple GOMS style knowledge chunks can use to modify system operation. The cognitive subsystem processes procedural and metacognitive rules in an identical fashion, simplifying development. Metacognition can be utilized to improve how the system restarts interrupted processes. COGNET has been designed to easily transition from a cognitive task analysis of a highly structured domain to a working system.

A forgetting mechanism has been incorporated into the COGNET architecture (Zachary et al., 2004). The forgetting capabilities were modeled after the Human Operator Simulator work of Glenn et al. (1992)

and Lane et al. (1981). Trace-decay and rehearsal effects (see Section II.4.1) were combined with proactive interference (see Section II.4.2) to model human learning performance. Only elements located within a COGNET model's short-term memory can be forgotten by these mechanisms. Contradictory information can be deliberatively removed from both short-term and long-term memory (Zachary et al., 2004).

COGNET has been used to construct models of real-time human-computer interaction in multi-tasking environments (Zachary et al., 1998). Attention-switching has been explored in vehicle tracking (Ryder and Zachary, 1991), and COGNET models have been used to create improved computer interfaces (Zachary et al., 1992) for en-route air traffic control (Seamster et al., 1993) and telephone operator services (Ryder et al., 1998). The Australian air traffic control curriculum has been redesigned with results from a COGNET air traffic control model (Australian Civil Aviation Authority, 1994). COGNET has been incorporated into the Operational driven development approach for Cognitive Systems (Reichel et al., 2008) cognitive agent system. This system allows a remote operator to direct teams of unmanned aerial vehicles operating within military domains.

Robots operating in dynamic domains and assigned complex tasks may require a large collection of skills, knowledge, and capabilities. The use of a cognitive architecture may be required for designs to effectively incorporate forgetting into a system capable of parallel and diverse processing capabilities. Existing cognitive architectures span a wide range of design paradigms and have been tested across a diverse sampling of experimental domains. Distilling the critical features of each may lead to a better understanding of the requirements for synthetic forgetting and its applicability to robotic domains.

II.4 Forgetting

Robots and humans operating within complex, dynamic domains are often inundated with data and forgetting can play a critical role in the ability to achieve and maintain SA. Forgetting information can remove out-of-date information, increase the ease of recalling critical knowledge, and improve cognitive processing efficiency. Robotic systems that aim to realize human levels of SA may require the beneficial, but intricate effects of forgetting.

A long standing debate has raged regarding the form and mechanics of human forgetting (Roediger III, 2008). Two prominent theories exist, time-based decay and similarity-based interference, which appear to stand in stark contrast to one another (Jonides et al., 2008). The first, subscribes to the belief that the passage of time directly degrades items within memory, while the later postulates that accumulation of inter-item interference prevents items from being successfully recalled. Numerous models of each theory have been developed that appear to accurately model empirical evidence, but the developed analyses and their models have also sparked controversy (Jonides et al., 2008; Mueller and Krawitz, 2009; Altmann and Schunn, 2002; Sims and Gray, 2004; Wixted, 2004). The complexity of psychological testing of humans has driven a serious debate over the possibility of diverse arrays of confounds that can cloud the interpretations of results (Jonides et al., 2008; Oberauer, 2006; Nairne, 2002; Wixted, 2004). Further complicating matters, it has been postulated that some psychologists have misread and misinterpreted previous findings, resulting in the dismissal of valid ideas and the reluctance to accept new theories (Wixted, 2004; Altmann and Schunn, 2002).

These debates are further heightened by the multiple forms of human memory, which can be partitioned into two major categories (Tulving, 1990). Procedural memory is highly stable, non-symbolic knowledge without a truth value, forming stimulus-response pairings. Conversely, declarative memory is “knowledge that can be introspectively reasoned over without any overt behavioral response” (Tulving, 1990). Declarative memory groups two separate, but parallel memory types, episodic memory and semantic memory (Tulving, 1984). Episodic memory maintains a collection of individual episodes and events, including temporal-spatial relations. Semantic memory is a non-instance based memory involving “the acquisition, retention, and utilization of skills and knowledge that have to do with the world” (Tulving, 1984). Some models and analyses of forgetting are strongly dependent on a particular form of memory. Altmann and Gray (2002) stated that their functional decay model relies heavily on episodic memory representations, although Sims and Gray (2004) questioned this dependence.

Item representation, which may be influenced by types of memory, adds additional uncertainty regarding the nature of forgetting. Presently, at least two forms of storage have been considered within short-term memory, whole item storage and the binding of features (Jonides et al., 2008). Models of both trace-based decay and interference-based forgetting have been developed utilizing both short-term storage approaches.

It has been posited that working memory operates on a number of levels, implying different item recall difficulties (Jonides et al., 2008). The Oberauer model (Oberauer, 2006) presents three levels of focus, “the activated part of long-term memory”, “the region of direct access”, and “the focus of attention”. The first represents activated long-term memory currently residing in working memory. The second represents highly activated items that can be directly accessed, while the third represents the item currently under focus. At the lower level, items may have only a small probability of being recalled, but they will provide structured noise, facilitating the forgetting process (Jonides et al., 2008).

Recall and reperception can affect the ability to recall items from memory. The spacing effect (Melton, 1967; Dempster, 1988; Pavlik Jr. and Anderson, 2005; Callan and Schweighofer, 2010) is the psychological phenomena where spaced repetition of reperception results in greater recall ability than repeatedly observing an item within a short period of time. While one of the most reproducible phenomenon in experimental psychology (Dempster, 1988), a complete understanding is still lacking. Extensive work has been completed exploring the spacing effect and a number of theories have been crafted (Pavlik Jr. and Anderson, 2005; Cepeda et al., 2006; Callan and Schweighofer, 2010). The deficient-processing theory states repeated presentation of an item induces a feeling of knowing (Zechmeister and Shaughnessy, 1980), resulting in reduced maintenance via rehearsal. When presentations are further spaced apart, the item will often no longer be in working memory and the sense of knowledge is reduced, resulting in greater rehearsal being performed (Rundus, 1971; Cuddy and Jacoby, 1982; Callan and Schweighofer, 2010). The encoding variability theory or contextual fluctuation theory (Estes, 1955; Melton, 1967; Landauer, 1969; Glenberg, 1979; Raaijmakers, 2003; Callan and Schweighofer, 2010) posits that when items are perceived in a spaced fashion, an increasingly diverse set of associations to existing knowledge forms along with the item's associated context becoming more diverse, aiding subsequent cue-based retrieval.

II.4.1 Time-based Decay

Time-based decay is the intuitive concept that items within short-term memory deteriorate and eventually disappear due to the effects of time (Jonides et al., 2008). At perception, items are encoded in memory at a particular activation level, dictating ease of retrieval. As time passes, this activation level decreases, increasing the difficulty of item retrieval. To combat these effects, the memory system performs a refreshing strategy called rehearsal (Nairne, 2002). During this process, memory items are recalled in order to strengthening their activation levels (Nairne, 2002). As shown in Figure II.12, the activation level of a memory item may undergo three stages (Altmann, 2002). *Strengthening* involves memory rapidly recalling the item to boost its activation level. The middle stage, *Use*, involves recalling a memory item to complete a task. During this period, recall provides a boost to the item's activation level, but not at a rate to fully counteract decay. In the final stage, *Disuse*, the item is no longer used and the activation slowly decays.

In 1885, Ebbinghaus (1885) presented the original forgetting curve, a logarithmic function, which can be estimated with a power function (Wixted and Carpenter, 2007), that predicts trace-decay (Roediger III, 2008). A commonly used equation for modeling time-based decay, Equation II.2, can be found in the ACT-R cognitive architecture (Anderson and Lebiere, 1998), see Section II.3.5.

While mathematical models of time-based decay exist (Mueller and Krawitz, 2009), there are no generally agreed upon biological mechanisms to support time-based decay, although several mechanisms have been proposed (Jonides et al., 2008). One suggestion postulates that neurons forming an item in memory fall out of synchrony, continually increasing the difficultly of retrieval (Lustig et al., 2005). A second states items do not deteriorate, but their probability of receiving memory's focus of attention wanes, making the item harder

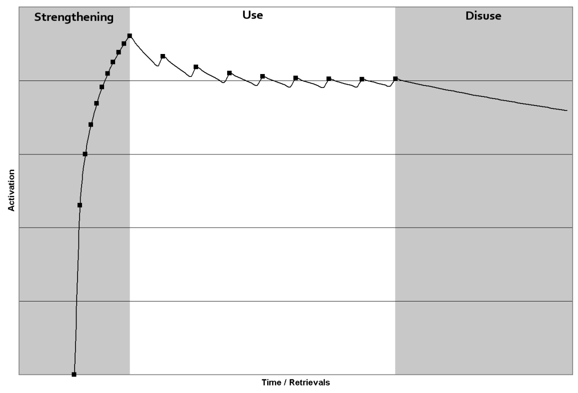


Figure II.12: Time-based decay (adapted from (Altmann, 2002))

to recall (Jonides et al., 2008).

Both psychological-based and neurological-based empirical evidence has been collected in order to verify time-based decay, but alternate explanations exist for many of the discovered trends (Jonides et al., 2008). These alternate explanations have been used as support for interference-based forgetting (Lewandowsky et al., 2004).

II.4.2 Similarity-based Interference

Similarity-based interference, the current dominant account of forgetting (Jonides et al., 2008), has a long and storied history (Wixted, 2004). While proponents of interference admit that forgetting is correlated with time, they believe the true causes of failed recall are processes and activities that occur during a time span (Nairne, 2002). Unlike decay theory, interference predicts failed recall results from inter-item competition for the memory system's focus of attention (Roediger III, 2008). Interference-based forgetting is a complex theory involving many forms of competition between items. Competition can be affected by item encoding strength, the number of items, the similarity between items, and the phase of learning and recall (encoding, storage, and retrieval) (Jonides et al., 2008). Two common methods exist for partitioning the theories of interference, the point of interference (encoding and output) (Lewandowsky et al., 2004) and the age of affected items (retroactive interference and proactive interference) (Jonides et al., 2008). The former compares Encoding Interference with Output Interference, while the later contrasts Retroactive Interference with Proactive Interference. Each form of interference exhibits different behaviors and many models of forgetting promote one over the other.

II.4.2.1 Proactive Interference

Proactive interference effects entail existing memory items decreasing the encoding strength of new items and increasing the difficulty associated with new item recall (Jonides et al., 2008). Proactive interference is often associated with cue-overload, a phenomena where items associated with the same or similar cues interfere with each other (Wixted, 2004). When a new item is initially perceived, the effects of proactive interference may be small, but will increase with time. It is speculated that items previously associated with a cue are unintentionally recalled during the new item's retention period resulting in interference (Wixted, 2004).

II.4.2.2 Retroactive Interference

Retroactive effects describe the phenomena where new memory items interfere with existing items (Jonides et al., 2008). While an object of debate, there appears to be a Retroactive Interference gradient immediately following the perception of an item. Some believe that after memory trace creation, a consolidation process begins that strengthens the item. As items consolidate, they become more resistant to interference and the effects of retroactive interference decrease (Wixted, 2004).

Skaggs (1933) has posited that retroactive interference effects actually arise from two separate processes, similarity and mental exertion. Receiving grater acceptance, similarity effects dictate that increased similarity between items results in greater retroactive interference (Jonides et al., 2008). Mental exertion provides non-similarity based retroactive interference. During an item's consolidation process, the presence of mental exertion of any form will adversely affect the consolidation process and the item's final encoding strength (Wixted, 2004).

II.4.3 Cognitive Models Incorporating Forgetting

Through the years, strong evidence has been found to both support and refute the two leading theories of forgetting, time-based decay and interference (Jonides et al., 2008). Some research has posited that forgetting is a combination of both theories and their subcomponents (Jonides et al., 2008; Wixted, 2004; Sims and Gray, 2004; Altmann and Gray, 2002). Reflecting on the complexity of the human mind and cognition, it is unlikely that the full dynamic of forgetting can be realized from one parsimonious mechanism. Several researchers have developed models of forgetting that incorporate multiple forgetting methods (Mueller and Krawitz, 2009). Others that stand by one mechanism, have admitted that multiple mechanisms may contribute to forgetting, just to a lesser extent (Wixted, 2004). Table II.5 presents a number of existing models of short-term memory and characterizes them based on the forms of forgetting that they contain. This table has been modified and expanded from (Mueller and Krawitz, 2009). As presented in (Mueller and Krawitz, 2009), the Time Effects column was labeled Decay and was changed to prevent confusion with purely decay-based

Table II.5: Models of short-term memory (Adapted and Expanded from (Mueller and Krawitz, 2009))

Model	Time Effects	Capacity	Encoding Interference	Output Interference
Perturbation Model (Lee and Estes, 1977, 1981)	+	-	-	-
(Shiffrin and Cook, 1978)	+	+	-	-
The Trace Threshold Model (Schweickert and Boruff, 1986)	+	-	-	-
TODAM (Lewandowsky and Murdock, 1989)	-	+	+	o
Network Model (Burgess and Hitch, 1992, 1999)	+	+	+	+
Feature Model (Neath and Nairne, 1995)	-	+	+	-
(Brown and Hulme, 1995)	o	-	o	o
ACT-R (Anderson and Matessa, 1997)	+	+	+	-
(Dosher and Ma, 1998; Dosher, 1999)	+	-	-	+
The Primacy Model (Page and Norris, 1998)	+	+	+	-
Start-End Model (Henson, 1998)	+	+	+	+
SPAN (Byrne, 1998)	+	+	+	-
EPIC (Kieras et al., 1999)	+	-	-	-
OSCAR (Brown et al., 2000)	-	+	+	+
Modified EPIC (Mueller, 2002)	o	o	o	o
SOB (Farrell and Lewandowsky, 2002)	-	+	+	+
Modified SOAR (Nuxoll et al., 2004)	+	-	-	-
SIMPLE (Brown et al., 2007)	+	-	+	-

'+' the component is present, 'o' the component is optional, '-' the component is not present.

forgetting effects. Table II.5 was expanded with the incorporation of Modified Soar. The remainder of this section will present a few models of forgetting that are representative of trace-decay and interference-based mechanisms.

II.4.3.1 The Trace Threshold Model

Schweickert and Boruff (1986) developed a model of trace-decay within short-term memory to explore verbal capacity limits. Unlike many models that promote short-term memory capacity limits, this model is designed around the length of time a verbal trace can be recalled. The model is founded on the idea that the "similarity between the trace-decay hypothesis and the common assumption made in psychophysics that the probability of detecting a stimulus equals the probability that its intensity is greater than a fluctuating threshold" (Schweickert and Boruff, 1986). Equations II.7- II.9 describe the entirety of the Schweikert-Boruff model. In this model, list recall will be successful if the verbal trace duration, T_v, is greater than the required recall time, T_r. Equation II.7 calculates the probability of correctly recalling a list. If T_r and T_v are considered to be stochastically independent random variables possessing means of τ_r and τ_v with variances of σ_r^2 and σ_v^2, equation II.7 can be transformed into equations II.8 and II.9. Equation II.8 calculates the probability of immediate serial recall "analogous to the psychophysical function for probability of detection" (Schweickert and Boruff, 1986). τ_v and σ_v^2 are determined through estimation with empirical data (Schweickert and Boruff, 1986).

$$P = P\left[T_r \leqslant T_v\right] \tag{II.7}$$

$$P = P\left[\frac{T_r - T_v - (\tau_r - \tau_v)}{\sqrt{\sigma_r^2 + \sigma_v^2}} \leqslant z\right] \tag{II.8}$$

$$z = -(\tau_r - \tau_v) \Big/ \sqrt{\sigma_r^2 + \sigma_v^2} \tag{II.9}$$

The Trace Threshold model accurately models the relationship between speaking times and the probability of correct list recall (Mueller and Krawitz, 2009), although it only takes maintenance into account (Schweickert and Boruff, 1986). Other components of memory span, including word frequency (Hulme et al., 1997), have been tested with the Trace Threshold model (Mueller and Krawitz, 2009). Results from the Trace Threshold model have been misinterpreted in the past, since unlike other models, this model treats a list of words as a single item. The more common approach is to treat words as individual items (Mueller and Krawitz, 2009).

II.4.3.2 SIMPLE

The Scale-Independent Memory, Perception, and Learning (SIMPLE) model (Brown et al., 2007) is a temporal difference model of human memory and forgetting. In this model, items to be recalled are located within a multi-dimensional psychological space. A temporal axis exists within this space that records the amount of time since an item's point of presentation and other axes can exist that record additional item properties. Since SIMPLE is a temporal difference model of forgetting, the discriminability of an item is determined by the degree to which the item is isolated from its neighbors. When calculating the discriminability of items, inter-item distance along the time-axis undergoes a logarithmic compression, that increases the effective similarity of items as they move further from the psychological space's origin. While time plays a critical role in the SIMPLE model, it does not incorporate trace-decay. Forgetting within SIMPLE is purely interference based. With Weber Compression, as time passes, items become increasingly similar and difficult to discriminate (Brown et al., 2007).

SIMPLE is a "scale-similar" model of forgetting, exhibiting similar behavior across time-scales differing by orders of magnitude (Brown et al., 2007). Similar behavior across time-scales is possible because of the way SIMPLE calculates inter-item similarity. Instead of comparing items by the magnitude of inter-item distance within psychological space, the ratios of positions along each axis, raised by a power constant, are compared. SIMPLE is also founded on the concept of local distinctiveness. As inter-item distance increases, the similarity between items decreases along with interference. Resultingly, items within close proximity affect each other to a greater extent than items spaced further apart (Brown et al., 2007).

The distance between two items in a multi-dimensional psychological space is computed with equation II.10. For each dimension, the difference is located along the relevant axis and scaled by a constant factor, w_l, representing the level of attention paid to that dimension. Distances in SIMPLE are calculated in a Manhattan fashion as opposed to an Euclidean or more complex method (Lewandowsky et al., 2004). The distance between two items is then processed by equation II.11 to form a similarity metric η. α modifies the family of equations used to transform distance into similarity. When α = 1.0, equation II.11 acts as an exponential function, while a value of 2.0 results in Gaussian behavior (Brown et al., 2007). Many experiments set α to a value of 1.0 (Brown et al., 2007; Lewandowsky et al., 2004). The discriminability of an item is inversely proportional to the similarity to the rest of the items in the system summed together, equation II.12. When recall omissions cannot occur, the probability of correctly recalling an item in a target location during serial recall is equivalent to the item's discriminability. However, the probability of incorrectly recalling an item at a desired location is given by equation II.13. In this equation, γ represents the deterministic nature of item recall. γ is often set to 1.0 (Brown et al., 2007). When modeling free recall, omissions are possible and equation II.14 is required instead of equation II.12. Equation II.14 transforms the probability density with a sigmoid function, governed by threshold value t and slope s.

$$S_{i,j} = \sum_{l=1}^{m} w_l |M_{i,l} - M_{j,l}| \quad \text{where} \sum_{l=1}^{m} w_l = 1 \tag{II.10}$$

$$\eta_{i,j} = e^{-c|S_{i,j}|^{\alpha}} \tag{II.11}$$

$$D_i = \frac{1}{\sum_{j=1}^{n} \eta_{i,j}} \tag{II.12}$$

$$P(R_j|T_i) = \frac{(\eta_{i,j})^{\gamma}}{\sum_{k=1}^{n} (\eta_{i,k})^{\gamma}} \tag{II.13}$$

$$P(R_i|D_i) = \frac{1}{1+e^{-s(D_i-t)}} \tag{II.14}$$

II.4.3.3 The Architecture of Dosher and Ma (1998) and Dosher (1999)

Dosher and Ma (1998) and Dosher (1999) developed a model of human forgetting that combines trace-decay with output interference effects. In this model of serial recall, trace-decay effects occur during explicitly labeled retention intervals and during the process of recall. As the length of time required to recall a list of items increases, forgetting effects become more severe and the probability of correct list recall decreases. Their model consists of two formulations, one predicting full list recall and a second that calculates individual item recall probabilities.

Equation II.15 determines the probability of correctly recalling an entire list during a serial recall trial and takes a common exponential form (Murdock, 1982; Wixted and Carpenter, 2007). Accurate list recall is influenced by list length (L), forgetting rate (β), and initial encoding strength of the entire list (λ). Depending on the form of experimental data being analyzed, t_L can represent output time, output duration, or pronunciation duration of the list. A Gaussian with mean $\mu'(t_L)$ and standard deviation σ models the strength distribution of the list during the forgetting process. Strength values above a threshold value τ are recalled, while lesser values indicate the list has been forgotten.

$$\begin{gathered} p = 1 - \Phi\left(\tau, \mu'\left(t_L\right), \sigma\right) \\ \mu'\left(t_L\right) = \lambda e^{-\beta t_L} \end{gathered} \tag{II.15}$$

Equation II.16 represents the probability of correctly recalling an individual item within a list of length l at its correct location (k) with material m and presentation rate r during serial recall trials. This equation incorporates two separate measures of time, the length of time between item presentation and the end of presentation (t_s) and the time duration between the end of list presentation and the point at which the item is to be recalled (t_o). Four scaling factors, β_{I_s}, β_{I_o}, β_{T_s}, and β_{T_o}, regulate the relative importance of the factors influencing forgetting. Item encoding strength $\alpha_k r$ is dependent on item position and presentation rate.

$$p = 1 - \Phi\left(\tau, \mu_{klmr}, 1\right) = \alpha_{kr} e^{-(\beta_{Is}(l-k)+\beta_{Io}(k-1)+\beta_{Ts}t_s+\beta_{To}t_o)} \tag{II.16}$$

Robotic systems have benefited from the introduction of ideas and concepts inspired by biological systems and humans in particular. Contemporary robotic designs are continuously incorporating additional human inspired capabilities. Human forgetting has the potential to motivate an analogous capability for robots. With increased forgetting abilities, future robots may be better equipped to realize benefits of forgetting similar to those currently enjoyed by humans.

II.4.3.4 Robotic Designs Employing Forgetting

Forgetting has previously been explored in the field of robotics. Kira and Arkin (2004) augmented an existing case-based reasoning system to use forgetting as a means of modifying a case library. Implemented within the MissionLab System (MacKenzie et al., 1997), an instance of the Autonomous Robot Architecture (Arkin, 1998), the modifications affected a behavioral unit called GOTO, which realizes goal-directed navigation in mobile robots. Cases in this system consisted of a fixed size collection of continuous valued vectors of static length, see Figure II.13. The system was able to learn new cases, but maintained a case library with a capacity that was quickly exceed. Each case was implemented as a fixed collection of scalar values representing internal parameters and traversability of areas directly adjacent to the robot (Likhachev and Arkin, 2001). Kira and Arkin's cases were fixed data structures, but many systems may use variable size constructs. Forgetting was implemented to aid in selecting cases the system could delete when its case library's capacity was exceeded.

To facilitate forgetting, three separate forms of case selection were evaluated: random selection, metric-based selection, and a spreading activation selection involving time-based decay. Random forgetting was not expected to produce quality results due to several properties inherent in the case-based reasoning system. The case library capacity was relatively small (four to twenty cases) and was expected to generally possess few redundant high performance cases. As the robot operated within its domain, existing cases were improved through a hill-climbing learning algorithm (Likhachev et al., 2002). Random selection primarily acted as a control group to compare other strategies. Four separate metrics were implemented in the metric-based forgetting mechanism, performance, recency, frequency of use, and a weighted summation of the three metrics. Activation spreading selection maintained activation scores for cases based on case construction, case performance, case similarity to the current environment, and case lifetime. Time effects resulted in a decrement to case activation similar to time-based decay, except the degradation was linear.

Testing occurred in the MissionLab simulator, with the robot navigating through both homogeneous and heterogeneous environments of various clutter density. Performance and frequency metrics best equipped the robot to complete the most tasks and minimize distance traveled. Weighting the metrics resulted in similar performance. Forgetting resulting from spreading activation performed worse than a robot not equipped with a forgetting mechanism. Recency and random forgetting achieved inferior results. Kira and Arkin attribute some of spreading activation's performance deficits to parameter and threshold settings.

While Kira and Arkin explored the benefits of forgetting for case-based reasoning, Correia and Abreu (2004) investigated improvements to topological map generation and maintenance from forgetting. In their work, fatigue and trace decay where incorporated into a behavior-based robot. This combination of abilities

Spatial Vector:		
D(goal distance) = 5		
	density	distance
Region 0:	σ_0 = 1.00	r_0 = 1.00
Region 1:	σ_1 = 0.80	r_1 = 1.00
Region 2:	σ_2 = 0.00	r_2 = 1.00
Region 3:	σ_3 = 0.80	r_3 = 1.00
Temporal Vector:		
(0 - min, 1 - max)		
ShortTerm_Motion R_s	= 0.000	
LongTerm_Motion R_l	= 0.600	
Traversability Vector:		
(0 - untraversable, 1 - excellent)		
$f_0 = 0.14$	$f_1 = 0.32$	
$f_2 = 1.00$	$f_3 = 0.32$	

Case Output Parameters:	
MoveToGoal_Gain	= 0.10
Noise_Gain	= 0.02
Noise_Persistence	= 10
Obstacle_Gain	= 0.80
Obstacle_Sphere	= 1.50
Bias_Vector_X	= -1.00
Bias_Vector_Y	= 0.70
Bias_Vector_Gain	= 0.70
CaseTime	= 2.0

Figure II.13: Example case-based reasoning case (Adapted from (Likhachev and Arkin, 2001))

was chosen to enable the robot to effectively manage the inter-play between exploration and preventing stale information from corrupting data stores. In their design, a simplified representation of fatigue was implemented as the latency, warm-up, and after-discharge portions of fatigue where assumed to have a zero second duration. Only the fatigue and recovery periods had non-zero durations. The robot possessed two behaviors, explore and go-home, of which explore had higher priority. In their experiments, the explore behavior would be enabled until the robot became fatigued or the map exceeded a threshold number of locations, causing the go-home behavior to be executed. After the recovery period, the robot would continue exploring the environment. Forgetting capabilities were used to purge stale information from the map. A trace-based decay approach was explored and three separate forgetting functions were evaluated, see equations II.17-II.19. In these equations, α represents the decay factor and n represents the number of times a location has been visited. The system was evaluated on simulated Kheprea robots (Correia and Abreu, 2004).

$$w(t+1) = w(t) - \alpha e^{\alpha t} \quad \text{where } w(t_0) = 1 \tag{II.17}$$

$$w(t) = \frac{1}{1+e^{-\alpha t}} \tag{II.18}$$

$$w(t) = 1 - \frac{1}{1+exp\left(-\frac{0.8t-3000}{500}+2.5n\right)} \tag{II.19}$$

Howell and Donald (2000) developed a robot that utilized forgetting while performing simultaneous localization and mapping tasks. During the execution of a task, the robot generated a map comprised of tangent vectors extracted from sonar readings. The number of data points within the map was capacity limited to a set number of entries and when this limit was exceeded, the system randomly discarded existing tangents. Removal of map entries provided benefits of bounding computational requirements by the mapping and localization algorithms and removing older information from the map. Howell and Donald (2000) presented three possible modifications to the forgetting mechanism that included restricting map evaluations to areas within close proximity to the robot, sampling available vectors to reduce the density of evaluated map entities, and performing localization at progressively more precise resolutions.

CHAPTER III

Detailed Presentation of Approach to Solving Problem

This chapter presents a novel approach to improving the data management capabilities of robotic systems through the incorporation of Human-Inspired Forgetting mechanisms. The developed forgetting system holds the potential to reduce the number of memory items requiring cognitive processing, while simultaneously improving a robot's ability to detect critical cues in the environment. Increasing the ability for robotic systems to wade through vast arrays of diverse data will not generate SA by itself, but will form one component that may combine with other components to form the interactions required for the realization of improved robotic SA.

Robotic SA is influenced by a wide array of components and processes, but data management forms an unifying theme connecting many of these factors. Effective data management may allow for better information filtering and efficient processing throughout a robotic system by providing structure, while allowing for improved robotic SA and performance. Forgetting may play an integral part in the ability for robotic systems to remove unnecessary and erroneous data, while enabling rapid recall of relevant memory items. A large body of work has been completed that attempts to model the inner workings of human memory and the presented approach takes inspiration from this previous work to develop a forgetting capability targeting the specific requirements of robotic systems.

III.1 Forgetting

Before a forgetting system can be implemented to improve a robot's SA, the specific benefits to SA must be determined. Each level of SA possesses different attributes that may be non-uniformly affected by forgetting mechanisms. Level one concepts are generally atomic, consisting of either sets of similar raw data items (e.g., logged sensor readings) or singular items receiving continuous updates (e.g., bump sensor status). Purging will have a minor impact on systems that only consider a sensor's current status, but may facilitate the pruning of archived raw data.

Level two SA may benefit the most from forgetting. Composite objects, level two constructs are the melding of multi-modal information to form higher level representations of possibly highly dynamic constructs. Purging may realize two separate benefits. First, level two items have relations to level one items. When these lower level items are altered or removed, level two items may be modified. Unnecessary relations and subcomponents may be deleted or entire items may be discarded. Second, the entire item may be deemed unnecessary and deleted.

Level three concepts may be affected in many of the same ways as level two concepts. The difference lies in the effects on the system. Level three concepts predict the effects of actions and the evolution of the environment. These items may already be highly dynamic and possess a short life-span. Additionally, level three concepts are less likely to be a subcomponent of another item. As a result, the effects of forgetting a level three concept may be slightly smaller than the forgetting of level two items.

Even limiting the scope of improving robotic SA to the advancement of data management leaves a significant frontier for advancement. The described forgetting mechanism may aid systems in removing erroneous data and pruning unnecessary information. Forgetting may appear to be an odd selection as a feature addition to a system predicated on comprehensive and complex data management. This belief primarily results because forgetting is often considered to be a negative property or activity, for example misplacing keys, forgetting items on shopping lists, and the inability to remember where a car was parked. However, forgetting actually provides many benefits to data management.

Forgetting is a critical cognitive process for humans operating within complex and changing environments. With forgetting, humans are able to improve the efficiency of memory retrieval. Selecting goal-relevant memories results in competition between memories, each vying for selection. It has been shown that humans use forgetting to progressively repress memories that are not selected, thus reducing the cognitive load on subsequent memory retrievals (Kuhl et al., 2007). Human forgetting can reduce "switch cost", the costs incurred when switching between tasks (Altmann, 2002). Through "within-run slowing", a performance degradation that occurs during the execution of a task, memories regarding the present task are reduced in encoding strength. This encoding reduction allows for easier replacement of task specific information with data relevant to new tasks (Altmann, 2002). Forgetting also aids in classification, a common and critical task in many real world domains. In many environments, categories undergo gradual, but significant change. Exemplar items may become faster or larger, technology may improve the properties of consumer devices, and cultures will evolve over time. With forgetting, humans are able to progressively reduce the influence of previous instances of categories stored in memory and incorporate new items, allowing humans to adapt to changing categories (Elliott and Anderson, 1995). Forgetting also aids heuristic inference. Heuristics play a critical role in humans' ability to handle complex decision problems under tight time constraints and demanding mental load by taking advantage of structure in the environment (Gigerenzer et al., 1999; Gigerenzer and Brighton, 2009). Under certain circumstances, cognitive capacity limitations have been shown to improve the ability for humans to detect correlations within data (Anderson et al., 2005; Kareev, 2005). However, it has been suggested that cognitive limitations promote the use of simpler detection mechanisms that can outperform more complex approaches in stable environments (Gaissmaier et al., 2006). The recognition heuristic states that when selecting between items, select the item that is recognized (Gigerenzer et al., 1999;

Goldstein and Gigerenzer, 2002). With forgetting, unimportant information is unavailable, thus preventing it from being recognized. Irrelevant information removal increases the chances that only important items are recognized, increasing the applicability and performance of the recognition heuristic (Schooler and Hertwig, 2005). Forgetting has also been shown to improve performance of the fluency heuristic (Schooler and Hertwig, 2005).

As described in Chapter II, there are several separate proposed means by which humans forget information. The present biological and psychological literature has suggested many forms of forgetting and combinations of those forms, but has not reached a unanimous decision as to which method or methods are actually used. As compelling evidence has been presented for many of the proposed methods, this dissertation introduces forgetting to robotic systems by selecting and combining several of these human forgetting methods into one comprehensive algorithm. Through the method that the individual components are combined, the relative importance of each forgetting component can be changed to realize a highly configurable and dynamic forgetting system.

III.2 Improving Situation Awareness with Forgetting

This dissertation recommends the incorporation of forgetting within robots in order to improve the generation and maintenance of human-level SA. Forgetting and recall mechanisms can be used to select memory items that have a high probability of solving a current problem or increasing the understanding of a situation. Forgetting can also be performed after a memory item has been selected. The recallability of the selected item is increased, while the ability to retrieve other items in memory is reduced. The approach taken by this dissertation pre-filters memory items, reducing the quantity available to selection mechanisms. A large percentage of existing and future robotic algorithms require substantial computational power, but must execute fast enough to realize a real-time system. Pre-filtering available memory items has the potential to not only remove out-dated, erroneous, and irrelevant data, but also to decrease the computational demands of current and future robotic technologies. All but the simplest forgetting algorithms will require some computational power, but when this demand is less than the demand required by existing robotic algorithms, the system may realize a net performance increase.

Robots are often assigned diverse tasks and asked to operate within varied domains and no single algorithm or approach will completely solve the entirety of robotic challenges. A large array of existing algorithms have already been developed that enable robots to move and operate within complex domains while achieving high levels of performance, but these approaches tend to be domain specific. No simple parsimonious algorithm may be capable of solving all of robotics' problems. The described forgetting algorithm will not replace any existing or future technologies, but may improve the performance of those systems and allow for more effective searches through the vast quantities of data that may be stored within robotic knowledge bases.

Different algorithms can be employed to pre-filter robotic data, but each will need to follow the same basic design. When memory items are perceived or otherwise created, their cognitive representations receive properties and statistics facilitating the forgetting process. Item reproception and recall results in the representations being appropriately updated to reflect current usage patterns. When existing robotic algorithms require the use of a memory item, the forgetting algorithm begins the pre-filtering process. The retrievability of each memory item is calculated and the resultant set of recallable items is presented to the querying system. In general, items are not permanently forgotten. The retrievability of an item is recalculated each time a query is to be performed. Depending on the forgetting algorithm used, the existence of memory items, even with a very small chance of recall, can affect the system similar to how cues can help humans recall information. Forgetting may not be appropriate for the entirety of robotic data and certain algorithms may need to bypass the filtering effects of forgetting. With the described approach, both of these cases can easily be handled. Forgetting will not have to be applied to every type of data within a robot. In situations where accuracy may be more important than execution speed or when a complete analysis of all memory items is required, forgetting pre-filtering can be ignored. The presented forgetting approach does not necessarily delete memory items determined to be unrecallable. In these situations, the full collection of memory items can be presented to the querying system.

III.3 ActSimple

A large number of models of human forgetting have previously been developed, but few models have been crafted explicitly for the control and implementation of mobile robots. This dissertation has developed a new method for realizing forgetting that has been developed specifically for robotic systems and the application of pre-filtering data available to existing and future robotic algorithms. The new algorithm, named ActSimple, derives its name from its two largest influences, ACT-R and SIMPLE. Many algorithms and models of human forgetting have been explored in the past, but ActSimple incorporates a breadth of mechanisms and theories that may provide the new algorithm with the diversity and flexibility to aid robotic systems across a wide range of domains, modalities, and tasks.

Through the years, strong evidence has been found to both support and refute the two leading theories of forgetting, time-based decay and interference (Jonides et al., 2008). Some research has posited that forgetting is a combination of both theories and their subcomponents (Jonides et al., 2008; Wixted, 2004; Sims and Gray, 2004; Altmann and Gray, 2002). Reflecting on the complexity of the human mind and cognition, it is unlikely that the full dynamic of forgetting can be realized from one parsimonious mechanism. Several researchers have developed models of forgetting that incorporate multiple forgetting methods (Mueller and Krawitz, 2009). Others that stand by one mechanism, have admitted that multiple mechanisms may contribute to forgetting, just to a lesser extent (Wixted, 2004).

ActSimple combines time-based decay, effects of mental exertion, similarity-based interference, and output interference. Through the unique collection of mechanisms in ActSimple, the algorithm may be highly amenable to the varied demands of robotic systems. The system may be tailored to specific domains and tasks by modifying constants and the relative importance of the individual mechanisms.

This section proceeds as follows, first the incorporation of time-based decay is explained, followed by similarity-based interference's impact on the system. Mental exertion effects are presented, followed by the influence of output interference. Finally, the integration of all of the components is explained. Constant scaling factors and offsets are represented by *c* symbols in the equations that follow.

III.3.1 Time-Based Decay

While time-based decay theory previously fell out of favor (Wixted, 2004), decay has recently regained a significant amount of support (Altmann and Schunn, 2002; Wixted, 2004). Time-based decay's durability and recent resurgence provides strong evidence that it has an effect on forgetting. Time effects must be incorporated into any effective robotic forgetting system. Purely interference based models of forgetting exist that incorporate aspects of time, such as SIMPLE (Section II.4.3.2), but their direct applicability to robotics is unclear. Unlike many laboratory-based recall experiments, robots will observe items multiple times and may require repeated recall of individual memory items. The SIMPLE model does not incorporate rehearsal, although it can model some rehearsal-based primacy effects. Brown et al. (2007) modeled several recall data sets by setting the effective time value used by the SIMPLE model to the time of last rehearsal. While this approach approximated the empirical data, the effects of previous item presentation and rehearsal were lost or minimized. Unlike some interference-based models, trace-based decay models can allow for a natural modeling of rehearsal effects.

$$Activation = ln(\frac{ItemRecallCount}{\sqrt{LifetimeOfItem}}) \tag{III.1}$$

While multiple forms for the algorithmic process of forgetting have been proposed that exhibit a decreasing proportional rate of decay (Wixted, 2004), the power law family of equations, which has been posited to best characterize forgetting (Wixted, 2004; Anderson and Schooler, 1991; Wickelgren, 1974), will be used. A commonly cited equation is the ACT-R base activation equation, Equation II.1, and its many simplified forms (e.g. Equation III.1). Pavlik and Anderson (2003) modified Equation II.1 to incorporate the item's activation level at each time of retrieval (Sims and Gray, 2004). This modification realizes the spacing effect of short-term memory (see Section II.4). The Pavlik and Anderson equation, Equation III.2, will be used to model time-based decay (a mental exertion scaling factor, φ, has been added for ActSimple). Depending on the robot, the initial base level activation, β, can be set to a fixed value or a dynamically generated value, such as a belief level (Thrun et al., 2005). $b_{i,j}$ represents the feature's activation level at the jth retrieval of item i and b_i represents the feature's base activation. r_j is the span of time since item i's jth recall. φ represents the effects of mental exertion, which are described in Section III.3.3.

$$\begin{gathered} d_{i,j} = c_1 e^{b_{i,j}} + c_2 \\ b_i = \varphi\beta_i + ln(\sum^{J_i} r_{i,j}^{-d_{i,j}}) \end{gathered} \tag{III.2}$$

III.3.2 Encoding Interference

Many complex domains require robots to possess diverse arrays of knowledge to achieve high performance levels. Developing SA requires acquiring data from many sources, which can quickly strain robotic memory capacities. Limits in raw storage and computational restrictions may further hamper SA generation. Incorporating encoding interference may aid robots in maintaining breadth of knowledge without purging or ignoring critical information. Two forms of encoding interference are incorporated into ActSimple, mental exertion and similarity-based encoding interference. Mental overload has been incorporated by different means than similarity-based encoding interference and will be described in Section III.3.3.

The similarity-based encoding interference component is based on the SIMPLE temporal ratio model of forgetting (Section II.4.3.2). Unlike the SIMPLE model; however, acquisition time of memory items is not used to calculate interference as perception and rehearsal-based time effects are already incorporated into the trace-based decay portion of the system. The difference effects of the SIMPLE model were integrated into ActSimple because of the inter-item interactions within psychological space that the model produces. Explicit knowledge items possess properties, either continuous or discrete, that provide inter-item differentiation. Exploiting these properties by locating memory items within a multidimensional psychological space, as performed by SIMPLE, may allow robots to approximate differences between data and maintain a wide breadth of knowledge. Maintaining separate axes within the psychological space, instead of generating and storing a single difference value, allows ActSimple to provide unique weights to the distances along individual axes and dynamically alter those weight values. Similarity-based encoding interference is calculated with three equations, Equation III.3, Equation III.4, and Equation III.5. δ represents a set of functions able to calculate the distance between two items on an axis within the psychological space. Functions were incorporated into the model instead of simple distances to allow for axes to be non-linear or contain subspaces (for instance the International Commission on Illumination 1931 Color Space (International Commission on Illumination, 1932)). Attention directed to each axis is indicated by θ, while Δ represents the full distance between two items in psychological space. Presently, the inter-item distance is calculated in a Manhattan Distance fashion, although Euclidean methods are possible. The similarity (η) of two items is calculated by an exponential function incorporating item confidence levels (γ), inter-item distance, and a factor (λ) that alters how distance is related to similarity. $\wp_i$ represents the discriminability of item i.

$$\Delta_{i,k} = \sum^{L} \theta_l \left|\delta_l\left(i,k\right)\right| \qquad \text{where } \sum^{L} \theta_l = 1 \tag{III.3}$$

$$\eta_{i,k} = \gamma_k e^{-c_3\left|\Delta_{i,k}\right|^{\lambda}} \tag{III.4}$$

$$\wp_i = \frac{\gamma_i}{\sum^{K} \eta_{i,k}} \tag{III.5}$$

III.3.3 Mental Exertion

It has been posited that memory items may consolidate or become more durable after they are perceived. Mental exertion during this consolidation period can increase the likelihood that an item will be forgotten (Wixted, 2004). This form of retroactive interference may be beneficial to robots operating in complex and dynamic environments, as it may foster a form of focused attention. When a robot is under a heavy computational load, the addition of more memory items may slow comprehension and subsequently reduce SA. By incorporating retroactive interference through mental exertion, irrelevant details may be encoded at reduced strengths and quickly removed. Conversely, important items will be recalled rapidly, boosting their encoding strength and negating the effects of mental exertion.

The effects of mental load have been incorporated into ActSimple by generating a scaling factor, φ, that modifies an item's initial base activation level. φ is computed by taking the integral of mental exertion experienced by the robot during a time-window near the item's perception, see Equation III.6. In this equation, t_0 and t_1 represent the bounds of the time-window. While in the time-window, the robot updates φ each time the item's encoding level is required.

$$\varphi = 1 - c_4 \int_{t_0}^{t_1} MentalExertion\, dt \tag{III.6}$$

III.3.4 Output Interference

When robots are operating within complex and dynamic domains, periods of low computational demand followed by intervals of high computational load will be a common occurrence. Output interference has been added to ActSimple in order to decrease the volume of data that must be processed during these high load periods. As a robot makes decisions, the effects of output interference increase, systematically removing irrelevant and possibly erroneous information as the rate of decision making increases. The greatest benefit from output interference may be seen in domains where each subsequent decision or action increases the importance of correct decision making, while reducing the available response time.

Inspired by Lewandowsky et al.'s model (2004), output interference has been incorporated into ActSimple. Lewandowsky et al. provided the SIMPLE model of human memory with output interference capabilities by modifying Equation II.11 into III.7, where o is a constant and n represents the number of output decisions that have already been completed plus one. This modification to SIMPLE motivated the addition of an output interference parameter (ϖ) to ActSimple, see Equation III.8. In this equation, ρ_j represents the amount of time since the j^{th} decision was made. Deviating from Lewandowsky et al.'s modification, ϖ was not incorporated into ActSimple's encoding interference component. Instead, the effects of output interference have been placed in the recallable probability equation (Equation III.10), described in Section III.3.5. ActSimple incorporates more components than SIMPLE and placing the output interference coefficient in the recallable probability equation allows the effects of output interference to affect the entire system.

Unlike the conditions where Lewandowsky et al.'s model was evaluated, robotic systems may be required to operate for long periods of time, making a significant number of recalls. With the original model, output interference never decreased and eventually the effects of output interference would prevent the robot from making memory retrievals. A power law decay, modeled after the power law decay properties of ACT-R, was added to the magnitude of the output interference scaling factor so that interference effects will wane during periods of infrequent recall.

$$\eta_{i,j} = e^{-c\left|S_{i,j}\right|^{\alpha} o^{n-1}} \tag{III.7}$$

$$\varpi = c_5^{c_6 + ln \sum^{J} \rho_j^{-c_7}} \tag{III.8}$$

III.3.5 Activation

Many of ActSimple's individual components are combined by Equation III.9 to form an activation level for each item in memory. Effects of trace decay and encoding interference are summed together with permanent (ε_1) and transient (ε_2) noise. Scaling constant c_8 allows the relative importance of trace decay and encoding interference to be altered.

$$\alpha_i = c_8 b_i + (1 - c_8)\wp_i + \varepsilon_1 + \varepsilon_2 \tag{III.9}$$

Forgetting within ActSimple is nondeterministic and uses a sigmoid function, shown in Equation III.10, to compute the probability that an item is available for recall. In Equation III.10, ς represents the slope of the sigmoid function. τ is a threshold on the activation values and Output interference (ϖ) scales the resultant recallability probability. Similar sigmoid-based probability functions have been used by existing cognitive architectures, such as ACT-R (Section II.3.5) and SIMPLE (Section II.4.3.2), but resultant probabilities are used differently in ActSimple. In architectures modeling human memory, resultant probabilities are often used to probabilistically select an item for recall. ActSimple uses item probabilities to determine if an item will be available for selection. This change allows specialized and domain specific robotic algorithms to continue to perform item selection. The forgetting mechanism only removes old and irrelevant information, speeding search, while adding a small degree of variability to the item selection process.

$$P(r_i|\alpha_i) = \left(\frac{\varpi}{\varpi + 1}\right)\frac{1}{1 + e^{-\varsigma(\alpha_i - \tau)}} \tag{III.10}$$

III.3.6 Retrieval Time

Some robotic algorithms may require or benefit from the ability to approximate the ease of memory item retrieval. The retrieval time equation from the ACT-R architecture (Section II.3.5) is incorporated into ActSimple to provide an estimate of a memory item's integrity and availability. A constant latency factor (ψ) scales the result of an exponential function on an item's activation level (α) to generate a retrieval metric. This metric can be used to model the latency required for an item to be recalled from memory.

$$\text{Retrieval Time}_i = \psi e^{c_9 \alpha_i} \tag{III.11}$$

While not directly part of the system, robots using this mechanism can utilize rehearsal strategies to modify the activation values of memory items. By periodically recalling particular items, the system will boost their activation levels, potentially increasing their focus levels and preventing the items from being filtered.

In complex and dynamic domains, robots must process large volumes of diverse data. Through the human forgetting inspired filtering and purging provided by ActSimple, it is believed that robots may be better equipped to generate and maintain SA.

CHAPTER IV

Test Domain

In order to explore the effectiveness of using Human-Inspired Forgetting to filter information, a simplified version of ActSimple was developed along with six other forgetting algorithms. A WiFi signal strength estimation domain was developed to empirically test these forgetting approaches. Simulated WiFi signal strength information was generated and provided to a simulated robot operating within a grid-based world. The forgetting algorithms were used to filter the database of stored signal strength readings, improving the overall estimate of available WiFi signal strength at any particular point within the environment.

Robots operating within complex and dynamic domains are frequently inundated with copious data readings and information. Much of this knowledge is redundant, erroneous, or rapidly becomes out of date. Human-inspired robotic forgetting mechanisms may be capable of partially alleviating this information processing problem. A large number of human forgetting models have been proposed by the psychology literature, many of which may benefit robotics. While the vast and varied array of possible robot tasks and information may make the selection of an optimal forgetting algorithm for robots impossible, several are more applicable to on-line implementations and the specific requirements of robots.

The experiments described in this dissertation were conducted to explore the effectiveness of several forms of forgetting as applied to a specific task and data type common to many robot domains. Modern mobile robots often require some form of wireless communication, with WiFi frequently providing a high bandwidth, low-cost communication medium. WiFi signal strength, and consequently communication reliability and speed, can be affected by many factors. Areas of minimal to no signal strength can be frequently encountered. In stable environments, WiFi hardware can often be carefully positioned to provide acceptable WiFi signal strength and avoid areas of no communication. Unfortunately, optimal placement of WiFi technology in complex and dynamic environments is frequently not possible. Many robotic deployments require careful multi-agent coordination; necessitating categorization and monitoring of WiFi capabilities within an environment. Some systems even utilize relocatable WiFi nodes to alleviate communication bottlenecks and areas of minimal signal strength (Basu and Redi, 2004; Reyes et al., 2006; Weiss et al., 2008).

The described experiments simulated a robot tasked with continuously exploring a grid-based environment to monitor and estimate WiFi signal strength. The robot collected WiFi signal strength samples which were used to generate an estimate of the available WiFi signal strength of the entire environment. Interpolation methods were implemented to allow for estimates where no sample readings were available. This task, along with being practical and frequently required, is representative of many activities performed by robots operating within complex and dynamic environments where low-level environmental properties can impact the probability of successful mission completion.

IV.1 Algorithms

Seven forgetting algorithms were evaluated to filter unimportant sensor readings: a simplified version of ActSimple, a trace-based decay method (labeled ACT-R), two similarity-based interference methods based on SIMPLE (referred to as SIMPLE and SIMPLE Update), two queue-based forgetting methods (labeled Queue-Dynamic and Queue-Static), and a random forgetting method. Testing was also performed without forgetting (referred to as No Forgetting) to act as a control for comparing forgetting algorithm performance.

ActSimple

The experiments used a simplified version of the ActSimple algorithm. Described by Equations IV.1, IV.2, IV.3, IV.4, and IV.5, the simplified version of ActSimple was used instead of the full version to better illustrate the core components of the algorithm (trace-based decay and similarity-based interference), minimize the need for parameter optimization, and to allow for a more direct comparison to the ACT-R and SIMPLE algorithms. The simplified version of the algorithm does not incorporate the Pavlik and Anderson (2003) modification, mental exertion, belief state effects, output interference, or noise. All remaining constants were considered adjustable parameters except for λ, which was fixed at 1.0. In the SIMPLE-inspired portion of the algorithm, the psychological space consisted of three axes: reading position along the x-axis, reading position along the y-axis, and WiFi signal strength.

Equation IV.1 is a simplified version of Equation III.2. The effects of mental exertion (φ) have been removed and the Pavlik and Anderson (2003) modification ($-d_{i,j}$) was replaced with a constant, d. Equations IV.3 and IV.4 are equivalent to Equations III.4 and III.5 except that the belief state effects (γ) have been removed. Equation IV.5 calculates the complete activation level for a memory item except, unlike Equation III.9, noise (ε_1 and ε_2) was not included. Mental exertion (Equation III.6) and output interference (Equation III.8) are not included in the simplified version of ActSimple. Equation IV.2 is unmodified from the full ActSimple Algorithm (Equations III.3).

$$b_i = \beta_i + ln(\sum^{J_i} r_{i,j}^{-c_d}) \tag{IV.1}$$

$$\Delta_{i,k} = \sum^{L} \theta_l \left|\delta_l\,(i,k)\right| \qquad \text{where } \sum^{L} \theta_l = 1 \tag{IV.2}$$

$$\eta_{i,k} = e^{-c_3|\Delta_{i,k}|} \tag{IV.3}$$

$$\wp_i = \frac{1}{\sum^{K} \eta_{i,k}} \tag{IV.4}$$

$$\alpha_i = c_8 b_i + (1 - c_8)\wp_i \tag{IV.5}$$

$$P_i = \frac{1}{1 + e^{-\varsigma(\alpha_i - \tau)}} \tag{IV.6}$$

Trace-based decay forgetting

The trace-based decay forgetting algorithm labeled ACT-R was implemented for these experiments using Equation IV.7 and Equation IV.8. Noise was not used by the implemented version of the algorithm. In Equation IV.8, ς is equivalent to the $\frac{1}{s}$ term from Equation II.3.

$$B_i = \beta + ln(\sum_{k=1}^{n} t_k^{-d}) \tag{IV.7}$$

$$P_i = \frac{1}{1 + e^{-\varsigma(B_i - \tau)}} \tag{IV.8}$$

SIMPLE and SIMPLE Update

The similarity-based interference forgetting algorithms, SIMPLE and SIMPLE Update, were inspired by the SIMPLE model of human memory. The psychological space used by these two algorithms possessed four axes: reading position along the x-axis, reading position along the y-axis, time, and WiFi signal strength. Both methods work in an identical fashion, except when a WiFi reading is reperceived (a WiFi signal strength value is collected multiple times at a particular location). When reperception occurs, the SIMPLE Update method updates the sample's time property to the current point in time and increments the perception count for the reading. Conversely, the SIMPLE algorithm increments the perception count, but does not update the perception time. Both forgetting algorithms where implemented as described in Section II.4.3.2.

Queue Dynamic and Queue Static

Two queue-based forgetting algorithms were explored, Queue Dynamic and Queue Static. Both algorithms possessed a fixed sized queue that stored WiFi signal strength readings. New readings were appended to the end of the queue, and once the queue capacity was exceeded, old readings were removed from the front of the queue. Similar to the differences between SIMPLE and SIMPLE Update, operation of the two queue methods deviated only when a reading was reperceived. If a reading was collected and it was already present in the queue, then the Queue Dynamic algorithm updates the readings perception count and moves the reading to the back of the queue. Queue Static also updates the readings perception count, but does not relocate the reading.

Random Forgetting

Similar in operation to the queue-based forgetting algorithms, a random forgetting algorithm was also tested in the experiments. This algorithm worked exactly like the Queue Static algorithm except when capacity was exceeded, readings were randomly selected for deletion. The newest reading was guaranteed to not be removed.

IV.2 Environment

The environment constructed for these experiments consisted of a 25 unit by 25 unit simulated grid world. Movements within the grid world were constrained to only horizontal or vertical transitions. All movements consisted of single-unit location changes, and the simulated robot moved at every time step unless the robot attempted to leave the bounds of the grid world. It was assumed that all sensing and data processing occurred within one time unit. WiFi sensor readings were taken at every time step, irrespective of whether or not the robot's position actually changed. WiFi signal strength within the environment ranged from 1 to 100 (weak to strong, respectively) and was determined by a location's proximity to WiFi basestations located within the environment, see Equation IV.9. WiFi signal strength values computed by Equation IV.9 were clipped to the range 1 to 100. For the experiments, the slope was set to 2.0 and the threshold remained at 4.0. Two basestation configurations were tested during the experiments, four basestations and eight basestations, see Figure IV.1. In the figure, strong WiFi signal strength is represented by darker areas, while lighter areas indicate weak signal strength.

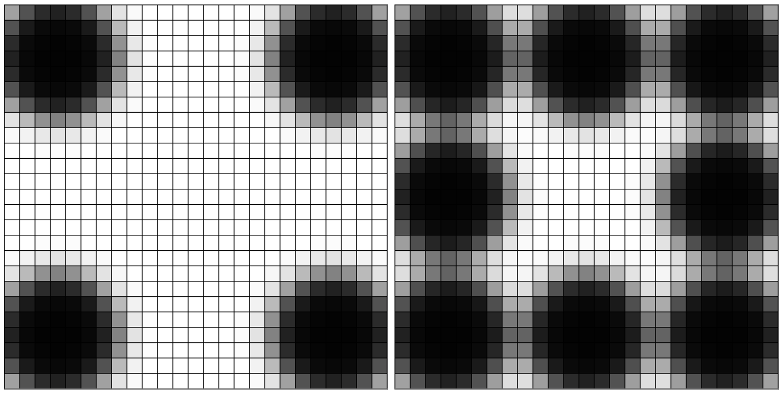

(a) Four basestation configuration (b) Eight basestation configuration

Figure IV.1: Basestation configurations within the tested environment

$$\text{WiFi Signal Strength} = 100 * \sum_i \left(1 - \frac{1}{1 + e^{-slope(distance_i - threshold)}}\right) + noise \tag{IV.9}$$

In addition to basestation proximity, WiFi signal strength readings were also influenced by noise. The magnitude of noise affecting a given sample was determined by a probability density function defined by Equation IV.10. Probabilities were not influenced by noiseless signal strength and the final reading value was clipped to remain within the interval 1 to 100.

$$p(x) = \begin{cases} 0.62 & \text{if } x = 0 \\ 0.1 & \text{if } |x| = 10 \\ 0.05 & \text{if } |x| = 20 \\ 0.03 & \text{if } |x| = 40 \\ 0.01 & \text{if } |x| = 60 \\ 0 & \text{else} \end{cases} \tag{IV.10}$$

While a grid world, the test environment possesses a significant challenge to a robot attempting to maintain an accurate estimate of WiFi signal strengths. The number of WiFi basestations can change, resulting in substantial alterations to the availability of WiFi signal strength, and noise will generate many erroneous readings. Even without changes to the number of basestations or noise, many naive approaches to estimating WiFi signal strength will not be effective. Figure IV.2 depicts the absolute error that would result if a robot simply assumed that the WiFi signal strength within a static, noiseless environment was constant. When four basestations are present, minimal error is realized when the robot estimates signal strength to be at its minimum, an approach that would be impractical and self-defeating. When eight basestations are present,

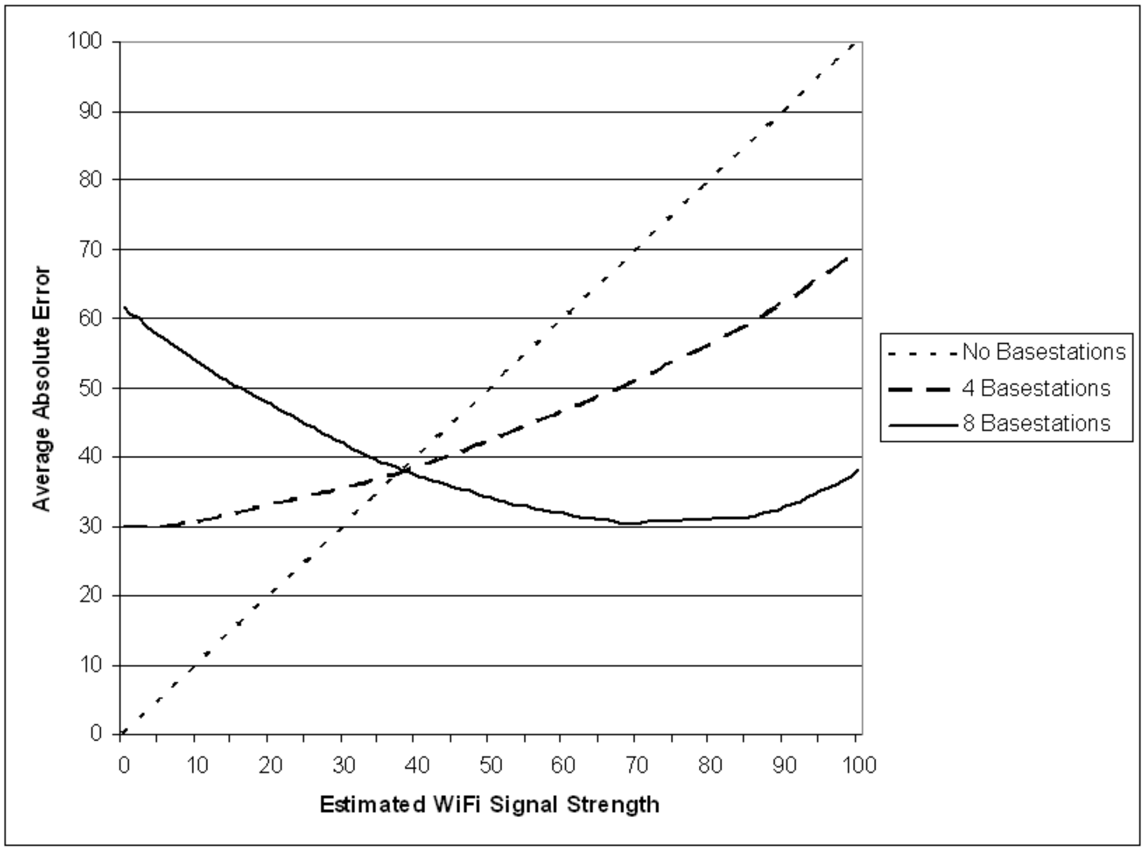


Figure IV.2: Average error when estimating WiFi signal strength with a constant value

the absolute error of a constant-valued estimate is radically different from when four basestations are located within the environment. A robot may require a large number of readings distributed within the environment in order to generate accurate estimates of WiFi signal strength. It is not possible to instantaneously acquire all of the required samples, since the process requires some period of time. Changing the number of basestations within the experiment models the dynamism present in nearly all real-world domains and tests the effectiveness of different data processing mechanisms' ability to handle real-world situations.

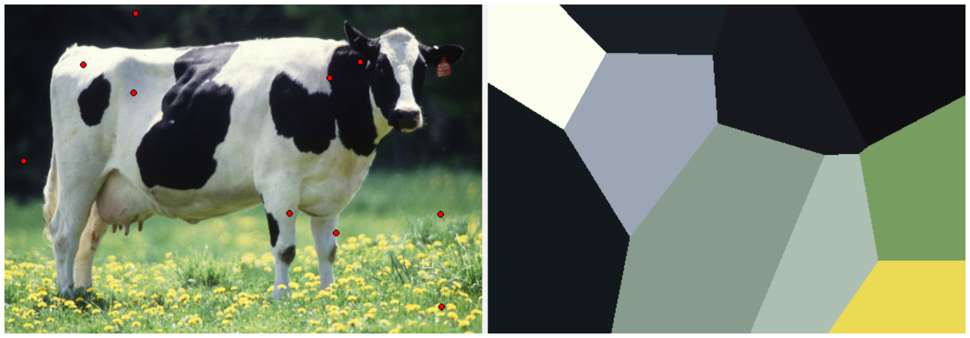
The effectiveness of the forgetting algorithms was evaluated by simulating a robot traveling along a number of predetermined paths taking WiFi sensor readings. At set intervals, the robot generated an estimated WiFi signal strength map from the readings that were correctly recalled. When each estimated WiFi signal strength map was created, the absolute error at each point within the environment was calculated and used as a metric for effectiveness. Interpolation methods were utilized to generate estimated signal strength values for locations where no sensor readings were successfully recalled. The absence of sensor readings at a given location may have resulted from the robot being unable to recall any readings at the location or the robot not having explored that area yet. When no readings were available, the robot guessed a strength value of 50.5, the midpoint of the range of possible signal strength values.

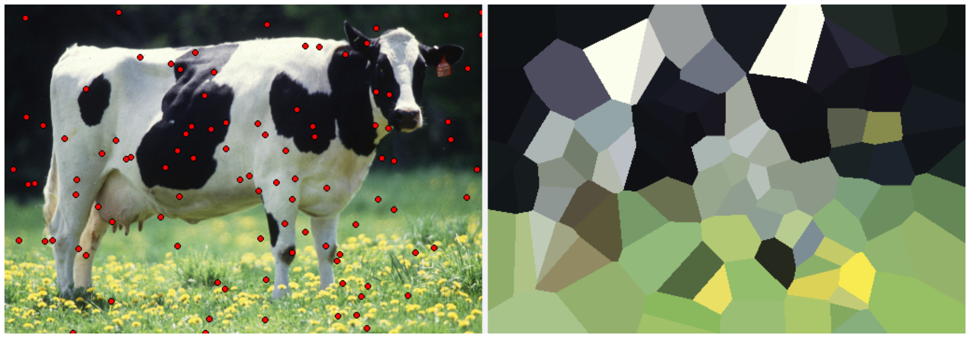
IV.3 Interpolation Methods

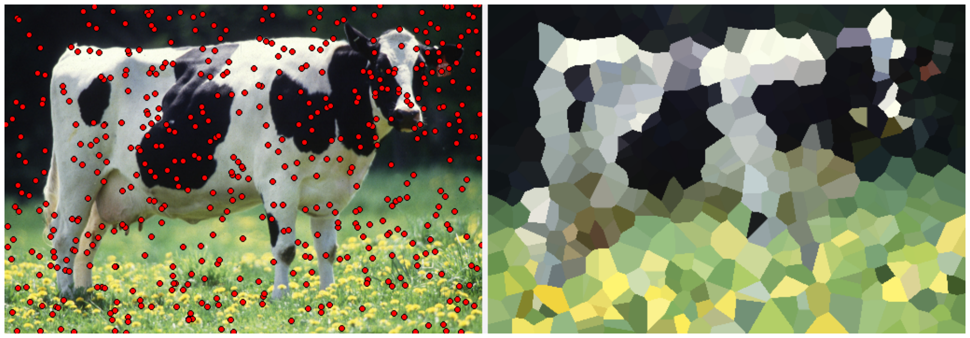
A large number of interpolation methods are available for converting the collection of sensor readings into an estimated strength map. The simulated environment was a grid-based world, but no restrictions were placed on the robot's path. Sampling was often non-uniform with regard to the entire environment and estimated strength maps needed to be created before the robot had a chance to fully explore the environment. The interpolation method used within the experiment needed to be non-uniform and allow for the generation of signal strength estimates outside of the convex hull surrounding the entirety of the presently recalled WiFi samples. A number of interpolation methods were possible, but the selection was further restricted to only include exact value methods of interpolation. Functional interpolation methods were not selected because, while estimated values may successfully pass through all provided sample points, these methods often generate invalid values for locations in between readings. With the nature of the experiments and sensor readings, these out-of-bounds values were unacceptable. Considering these criteria, the experiments used a combination of three exact interpolation methods: Nearest Neighbor Interpolation, Shepard's Interpolation (Shepard, 1968), and Microsphere Interpolation (Dudziak, 2007). Each of these forms of interpolation possesses a different method of generating estimated values that can affect the experimental results.

Nearest Neighbor Interpolation is a fast method that estimates values solely on the closest available sample. Due to the nature of the environment and the possibility of multiple readings being collocated, several WiFi readings may be equidistant to a location where interpolation is being performed. When equidistant points occur, two separate methods can be independently tested. The first approach uses an equal weighting scheme and averages the values for all samples (this approach will be refereed to as *Nearest Neighbor Interpolation with Equal Weighting*). The second approach uses a weighted average based on the number of times each reading has been observed (this approach will be referred to as *Nearest Neighbor Interpolation with Perception Count Weighting*). Nearest Neighbor Interpolation can be the most variable of the three interpolation methods. Small changes in the location of a reading may change which sample's value is used to estimate a target location within the environment. Since individual readings vary substantially, the estimate can equally fluctuate. Figure IV.3 demonstrates an example application of the Nearest Neighbor Interpolation method where the number of samples available to the interpolation algorithm is 10, 100, 500, and 1000 samples respectively. The left column shows the original images and the sample locations used by the interpolation method and the right column provides the resultant images. Nearest Neighbor Interpolation generates blocky results with few gradients.

The **Shepard's Interpolation** method (Shepard, 1968) was also explored during these experiments. An


(a) Nearest Neighbor Interpolation with 10 samples


(b) Nearest Neighbor Interpolation with 100 samples


(c) Nearest Neighbor Interpolation with 500 samples

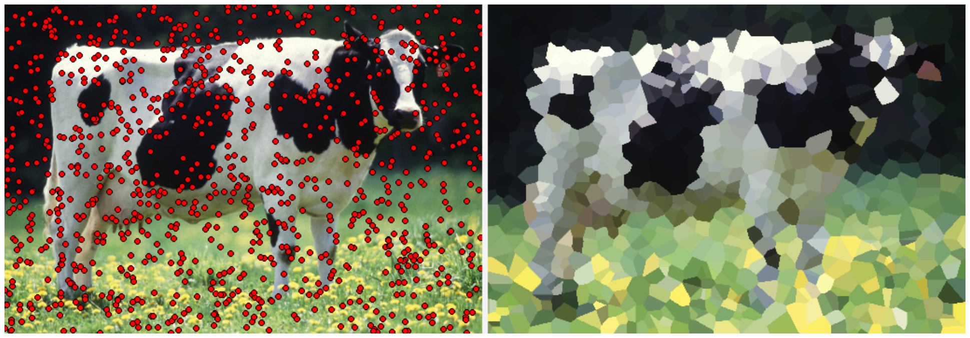

(d) Nearest Neighbor Interpolation with 1000 samples

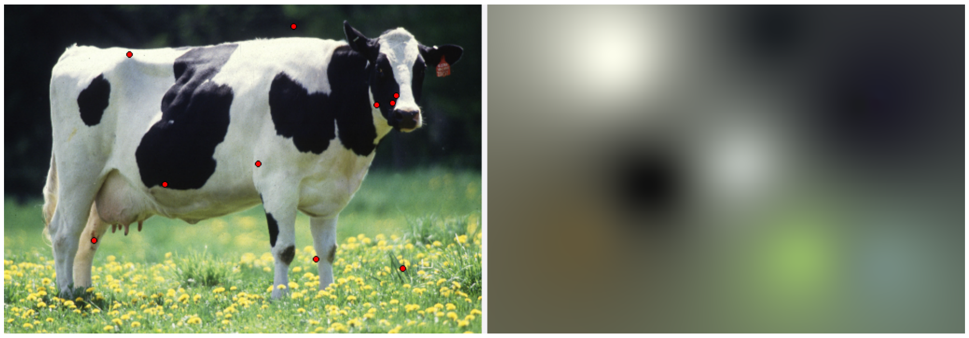
Figure IV.3: Nearest Neighbor Interpolation with 10, 100, 500, and 1000 randomly generated samples. The left column shows the original image and the sample locations used by the interpolation method and the right column shows the resultant images.

inverse distance weighted approach to interpolation, Shepard's Interpolation (Equations IV.11 and IV.12) differs from Nearest Neighbor Interpolation by incorporating influence from every available sample. Equation IV.11 calculates the weight or relative influence of sample i. p is the weighting exponent, a constant influencing propagation, while d represents the euclidean distance between a sample and the location that is being estimated. In the experiments, p was held constant at 2.0, a commonly used value for the weighting exponent (Dudziak, 2007). The sample weights are used in Equation IV.12 to generate a weighted sum of the values of available samples, v. The influence of individual readings is determined solely by relative distance between the samples and the location whose value is being interpolated; no special handling is required for equidistant points. An example of Shepard's Interpolation on a raster image is presented in Figure IV.4. In this figure, interpolation was performed with 10, 100, 500, and 1000 randomly generated samples. Shepard's Interpolation results are blended and not blocky like with Nearest Neighbor Interpolation; however, as shown in Figure IV.4, the results can be overly blurred, obscuring transition changes.

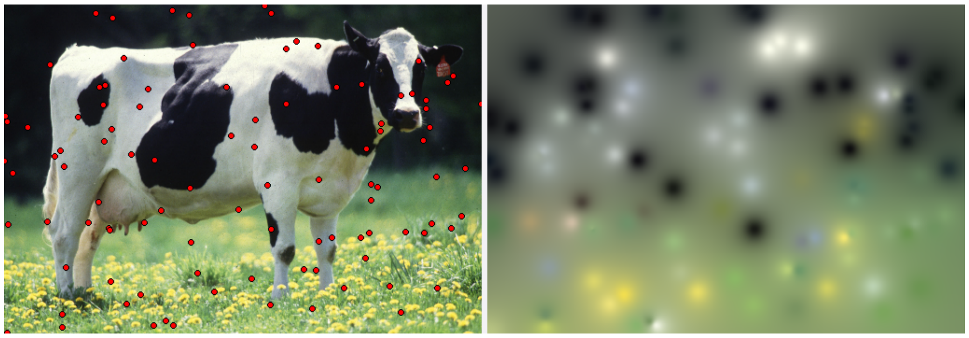
$$w_i = \frac{d_i^{-p}}{\sum^{N} d_j^{-p}} \tag{IV.11}$$

$$\text{Estimated Value} = \sum^{N} w_i v_i \tag{IV.12}$$

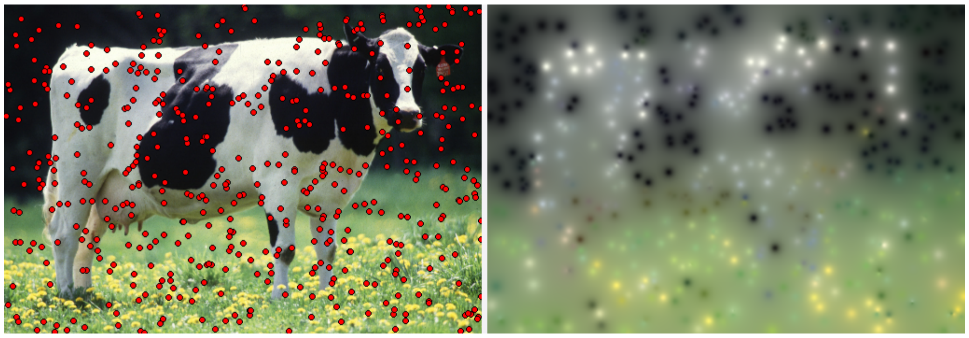
Microsphere Interpolation (Dudziak, 2007) is an approximation method embodying an analogy of spheres illuminated by light sources. The location being estimated is considered to be a sphere, with its surface decomposed into a number of sectors. Unlike in Shepard's Interpolation, the influence of a sample is affected not only by distance to the target location, but also by the angle between the sample and the normal from each sector on the sphere, see Figure IV.5. Similar to Nearest Neighbor Interpolation, the most influential sample for each sector is determined and then the samples selected for all of the sectors undergo a weighted average to produce the final interpolated value. The mathematical form of Microsphere Interpolation is shown in Equations IV.13, IV.14, and IV.15. The weighting exponent, p, in the Microsphere Interpolation algorithm serves a similar purpose as the weighting constant in Shepard's Interpolation. In the described experiments, when multiple samples equally affected a sector, their values were averaged with equal weighting. p was held constant at 2.0 and the number of sectors was set to 20. An example of Microsphere Interpolation on a raster image is presented in Figure IV.6. In this figure, interpolation was performed with 10, 100, 500, and 1000 randomly generated samples. Microsphere Interpolation combines the positive effects of Nearest Neighbor Interpolation and Shepard's Interpolation to produce blended results that still



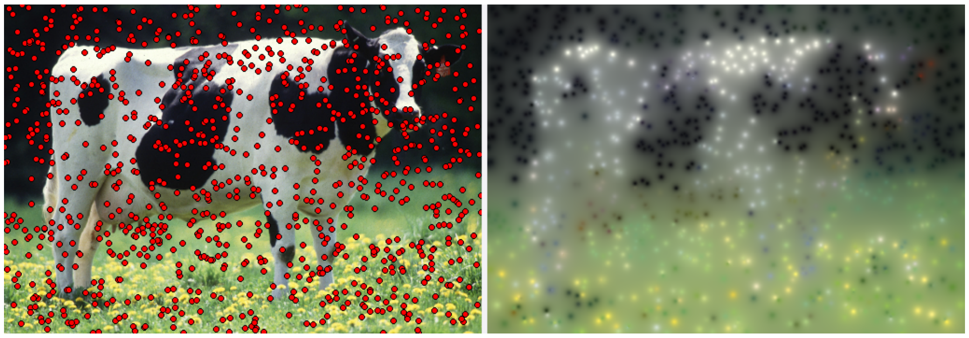
(a) Shepard's Interpolation with 10 samples



(b) Shepard's Interpolation with 100 samples



(c) Shepard's Interpolation with 500 samples



(d) Shepard's Interpolation with 1000 samples

Figure IV.4: Shepard's Interpolation with 10, 100, 500, and 1000 randomly generated samples. The left column shows the original images and the sample locations used by the interpolation method and the right column shows the resultant images.

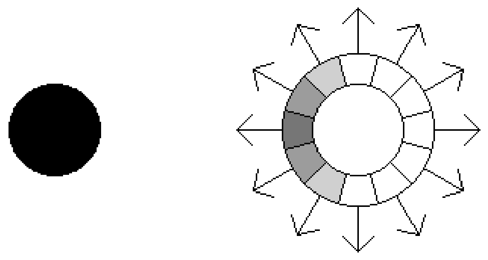


Figure IV.5: Microsphere Interpolation with one sample (adapted from (Dudziak, 2007))

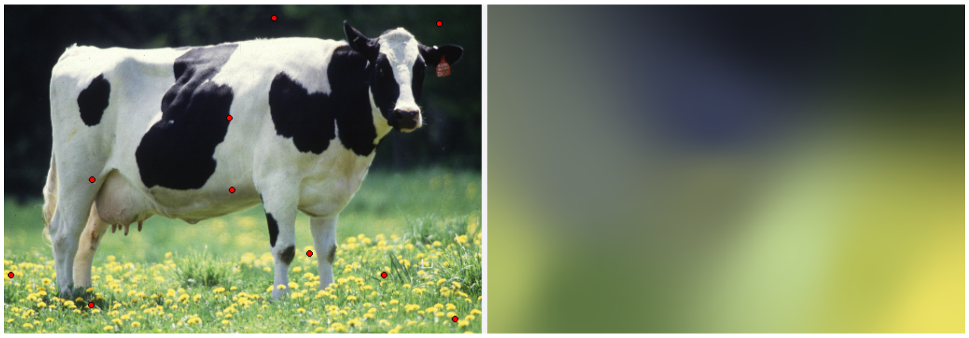
maintain a majority of the transitional information in the data.

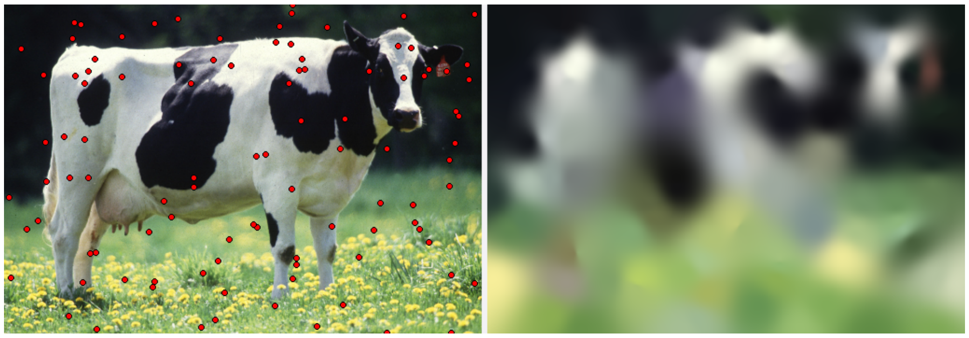
$$w_i = max\left(\left|l_j - I\right|^{-p} * cos\left(s_i, l_j - I\right)\right) | j \in (1,2,...,N) \tag{IV.13}$$

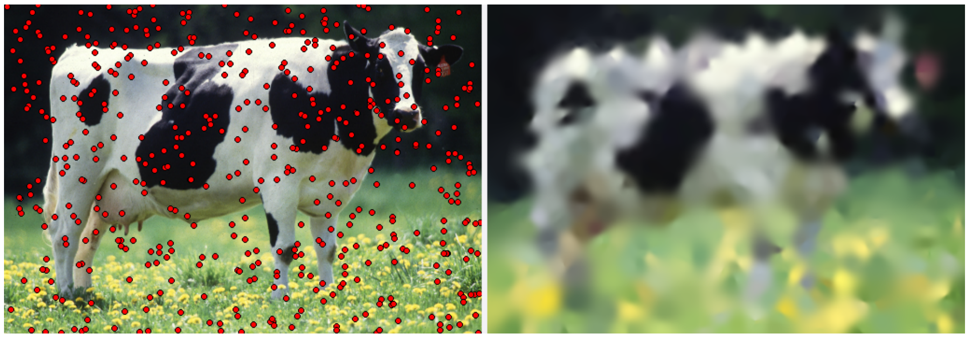
$$m_i = v_j | max\left(\left|l_j - I\right|^{-p} cos\left(s_i, l_j - I\right)\right), j \in (1,2,...,N) \tag{IV.14}$$

$$\begin{gathered} f(I) = \begin{cases} v_i \, | I = l_i, i \in \{1,2,..,N\} \\ \dfrac{\sum\limits_{i=1}^{P} m_i w_i}{\sum\limits_{i=1}^{P} w_i} & \text{otherwise} \end{cases} \\ I = \text{Location of interpolation} \\ p = \text{Propagation of influence power, } p > 0 \\ v_i = \text{Value of sample } i\text{, } i \in \{1,2,..,N\} \\ l_i = \text{Location of sample } i\text{, } i \in \{1,2,..,N\} \\ N = \text{Number of samples} \\ s_i = \text{Sector normal } i\text{, } i \in \{1,2,..,N\} \\ P = \text{Precision (Number of sector normals)} \end{gathered} \tag{IV.15}$$

The interpolation values of the described interpolation methods may have an impact on a robot estimating WiFi signal strength. Figure IV.3 shows the blocky nature of the results of Nearest Neighbor Interpolation. The edges of color boundaries are not approximated well and small variations in the samples can have large affects on the resultant interpolation values. Shepard's Interpolation does not have sharp transitions between color regions, but the results are overly smooth. The influence of distant samples are used in the generation of the interpolated value, generating incorrect results. These effects are most noticeable in the top center portion of Figure IV.4. Microsphere Interpolation limits the stochastic nature of Nearest Neighbor Interpolation,


(a) Microsphere Interpolation with 10 samples


(b) Microsphere Interpolation with 100 samples


(c) Microsphere Interpolation with 500 samples

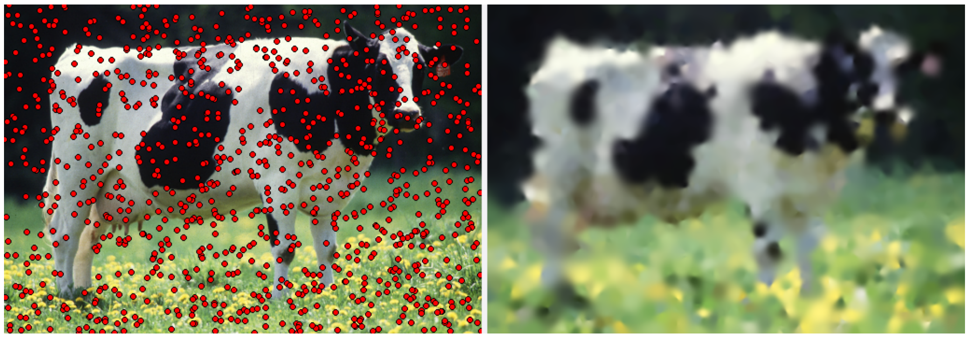

(d) Microsphere Interpolation with 1000 samples

Figure IV.6: Microsphere Interpolation with 10, 100, 500, and 1000 randomly generated samples. The left column shows the original images and the sample locations used by the interpolation method and the right column shows the resultant images.

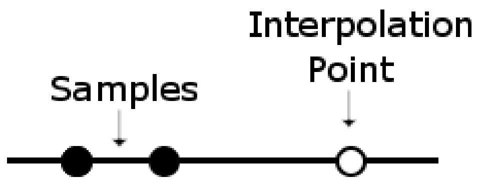


Figure IV.7: Multiple samples on the same line as an interpolation point

but also limits the influence of distance samples. When interpolating WiFi signal strength, distant readings should be ignored, as closer samples make their influence undesirable, especially if those points are collinear with the location to be interpolated. As shown in Figure IV.7, if two readings, that are collinear with the interpolation location, are on the same side of a location to be interpolated, the more distant point (assuming no error), should not be used in the interpolation. The two samples may have completely different values, possibly resulting from objects within the environment, and averaging will only degrade the quality of the result.

The form of weighting used when readings are equidistant can have a large impact on the quality of interpolation results. When asymmetric noise is present within a system, perception count weighting may be better equipped to minimize average error as the number of points available to the interpolation method increases. Figure IV.8 shows the results of the four described interpolation methods on data collected from the experimental environment. Figure IV.9 displays a zoomed in view of the first ten thousand steps taken by the robot. The noise within the environment is asymmetric because of WiFi signal strength clipping. The data shown in the figures was gathered by a simulated robot, without forgetting, that continuously explored the entirety of the experimental environment while collecting WiFi signal strength readings. All four interpolation methods initially generated a high level of estimation error, but as the robot started collecting readings, the estimation error quickly decreased. After approximately 500 time-steps, the effects of perception count weighting and equal weighting diverged. Nearest Neighbor with Equal Weighting, Shepard's Interpolation, and Microsphere Interpolation all experienced increasing error, while Nearest Neighbor with Perception Count Weighting generated a reduction in error as the number of steps increased. Asymmetric noise causes this phenomena. When noise is symmetric, highly erroneous readings will effectively average out; reducing error over time. Asymmetric noise; however, does not generate complimentary erroneous data readings and the average of all sensible reading values do not average to the correct value. Perception Count Weighting is only able to average out noise when multiple readings have been acquired for a given location. Depending on the paths taken by robotic systems within an environment, the availability of multiple readings may be inconsistent.

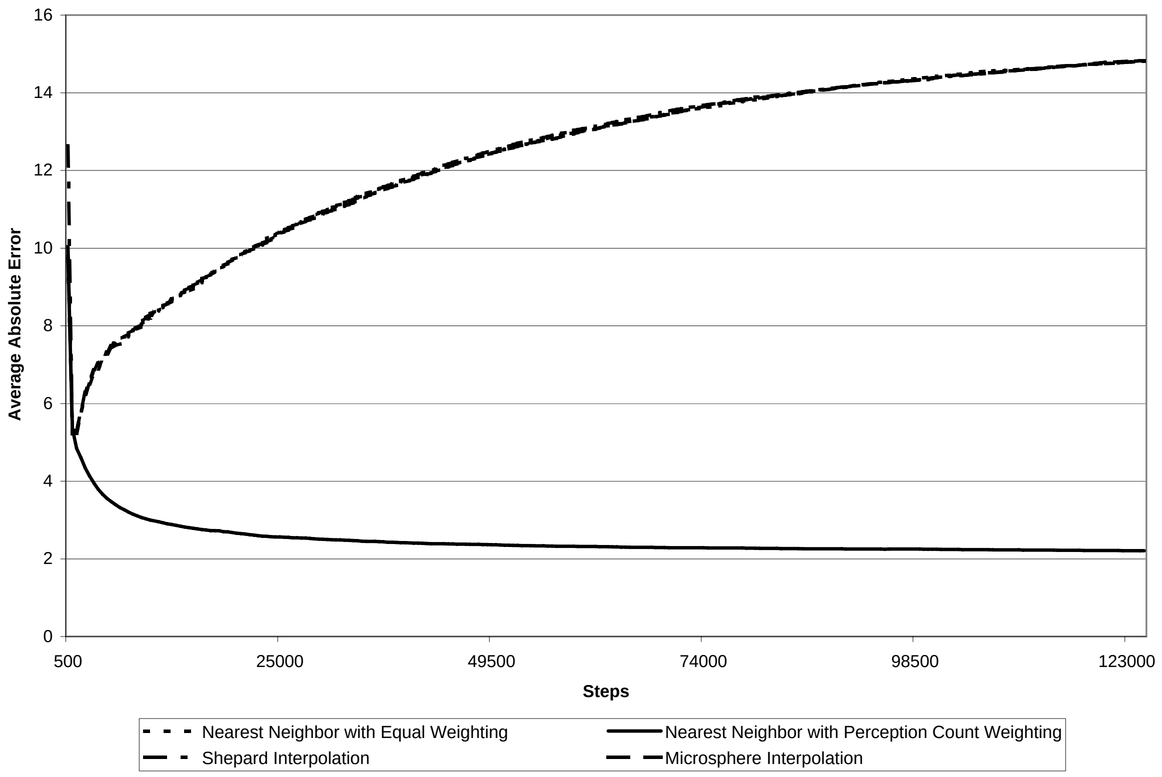


Figure IV.8: Effects of asymmetric noise on interpolation methods

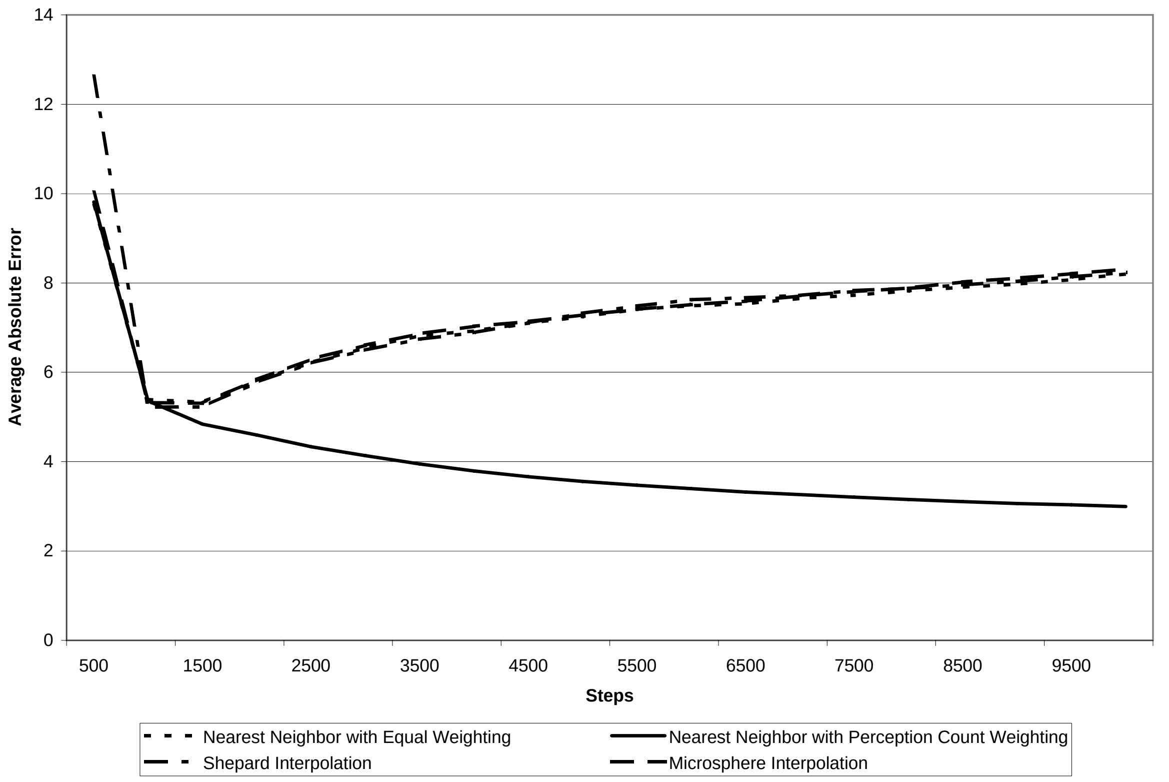
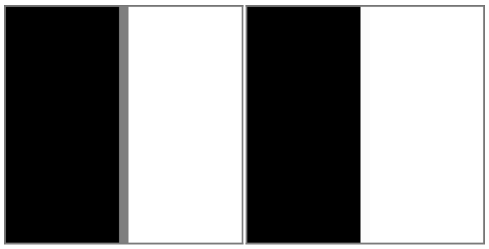


Figure IV.9: Effects of asymmetric noise on interpolation methods for the first ten thousand steps taken

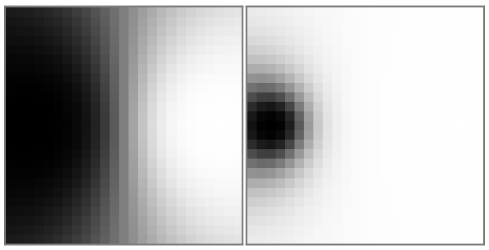
Robots operating within complex domains frequently do not evenly sample an environment. Task execution, terrain traversability, and other entities can influence the paths that robots follow and the length of time that they spend in various locations. Especially when sensors are continuously acquiring new samples, clustering and grouping of data generally occurs. Nearest Neighbor Interpolation, Shepard's Interpolation, and Microsphere Interpolation are all affected differently by data clustering. Figure IV.10 demonstrates the effects of clustering on the three interpolation methods. The images were constructed by interpolating across a 25 pixel by 25 pixel image. A black pixel sample was located at the point (-10, 0) and 1 or 100 white pixel samples were located at the point (10, 0). The one hundred white pixels were all collocated. The results of one black pixel and one white pixel are shown in the left column of Figure IV.10 and the results of one black pixel and one hundred white pixels are shown in the right column. Nearest Neighbor Interpolation (Figure IV.10a) is only affected by data clustering when samples are equidistant, resulting in a weighted average. Shepard's Interpolation (Figure IV.10b) is affected at all points within the image, except at the exact locations of the samples. The presence of one hundred white pixel samples dramatically influences the interpolation values throughout when using Shepard's Interpolation (Figure IV.10b). Microsphere Interpolation (Figure IV.10c) generates identical results in either situation. Mobile robots are tasked with varied and diverse assignments with many data modalities. The data readings resulting from these modalities may resemble any of the three interpolation methods. An experiment presented by this dissertation explored the effects of all three interpolation methods to evaluate the effects of clustering on the effectiveness of forgetting in mobile robotics.

IV.4 Experimental Design

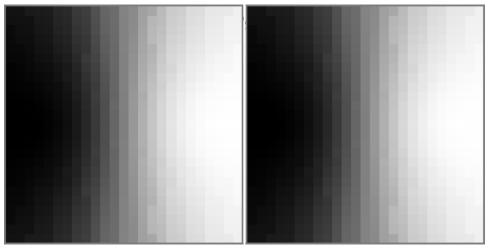
Robotic forgetting of low-level data readings may be affected by data clustering, reading distributions, and environment coverage. In the described experiments, the results of a simulated robot following twenty separate paths through the environment were tested. These paths, graphically described in Figure IV.11, fit into two different categories. The first sixteen paths, Figure IV.11a-p, involve the robot systematically moving back and forth through the environment, completing a number of cycles. A cycle refers to the simulated robot traveling from the left portion of the environment to the right and then completing a return trip. The remaining four paths, Figure IV.11q-t, were randomly generated. While the robot traversed the paths, either four or eight WiFi basestations were present, see Figure IV.1. In some of the paths, the number of basestations changed from four to eight or from eight to four at roughly the half way point through the complete path. These changes were incorporated into the testing in order to evaluate the system's ability to adapt to changing environments. At the completion of each path, the robot used an interpolation algorithm to generate an estimated WiFi signal strength map. The absolute error between the true WiFi signal strength and the



(a) Nearest Neighbor Interpolation



(b) Shepard's Interpolation



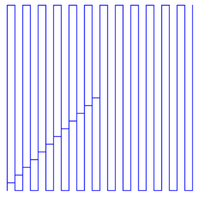
(c) Microsphere Interpolation

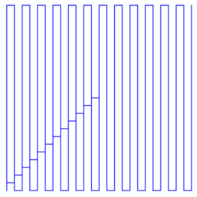
Figure IV.10: Effects of sample clustering on interpolation methods

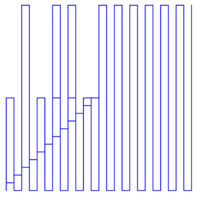
estimate was calculated for each location within the environment and averaged. This mean value formed the primary metric to indicate performance of a particular forgetting algorithm (or no forgetting) on the recently traversed path.

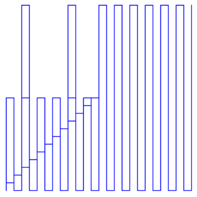
IV.5 Summary

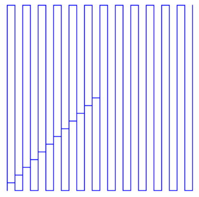
This chapter described the general environment and testing procedure used for all experiments presented in this dissertation. However, most of the experiments made modifications to this general design. Chapter V presents an experiment that computed optimal parameterizations for the seven forgetting algorithms. The algorithms, combined with the parameterizations, were tested with 44 new paths and four new basestation configurations. Chapter VI presents an experiment conducted in the real world (not in simulation). During this experiment, the simulation environment and the noise distribution described in Chapter IV.2 were not used. Chapter VII presents an experiment where the forgetting algorithms were tested in the presence of twenty one new noise distributions. An experiment is presented in Chapter VIII where the forgetting algorithms were tested at each point along the paths, not just at the end. Finally, Chapter IX presents an experiment where the Pavlik and Anderson modification (Chapter III.3.1) was incorporated into the tested versions of the ActSimple and ACT-R algorithms.

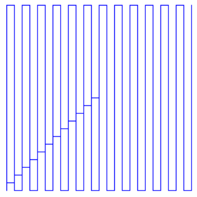

(a) Path 1
1 cycle
4 to 8 basestations
1272 steps

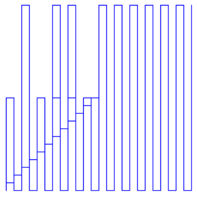

(b) Path 2
1 cycle
4 basestations
1272 steps

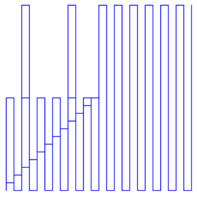

(c) Path 3
1 cycle
4 basestations
1080 steps

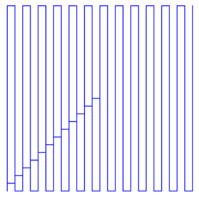

(d) Path 4
1 cycle
4 basestations
1032 steps

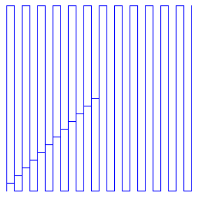

(e) Path 5
1 cycle
8 to 4 basestations
1272 steps

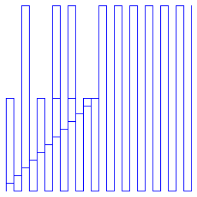

(f) Path 6
1 cycle
8 basestations
1272 steps

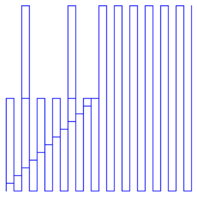

(g) Path 7
1 cycle
8 basestations
1080 steps

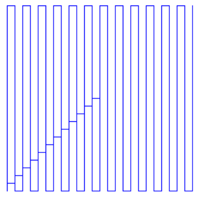

(h) Path 8
1 cycle
8 basestations
1032 steps

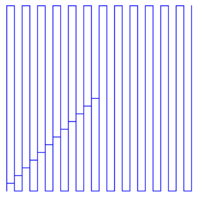

(i) Path 9
3 cycles
4 to 8 basestations
3768 steps

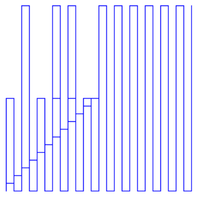

(j) Path 10
3 cycles
4 basestations
3768 steps

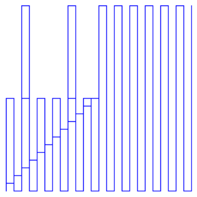

(k) Path 11
3 cycles
4 basestations
3192 steps

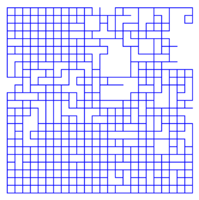

(l) Path 12
3 cycles
4 basestations
3000 steps

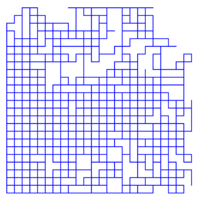

(m) Path 13
3 cycles
8 to 4 basestations
3768 steps

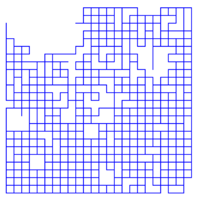

(n) Path 14
3 cycles
8 basestations
3768 steps


(o) Path 15
3 cycles
8 basestations
3192 steps


(p) Path 16
3 cycles
8 basestations
3000 steps


(q) Path 17
random path
4 to 8 basestations
3000 steps


(r) Path 18
random path
4 basestations
3000 steps


(s) Path 19
random path
8 to 4 basestations
3000 steps

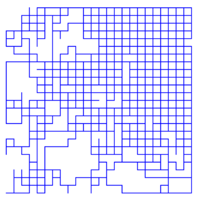

(t) Path 20
random path
8 basestations
3000 steps

Figure IV.11: Paths through the environment

CHAPTER V

Optimization

The forgetting algorithms described in Chapter IV.1 each possess a number of parameters that can greatly influence forgetting performance and estimation accuracy. Before the effectiveness of the forgetting algorithms were evaluated, the algorithms required optimization. This chapter describes the optimization process performed to identify the optimal parameterizations for each forgetting algorithm through the use of simulation. Using the calculated parameterizations, the relative performance of each forgetting algorithm and No Forgetting was analyzed.

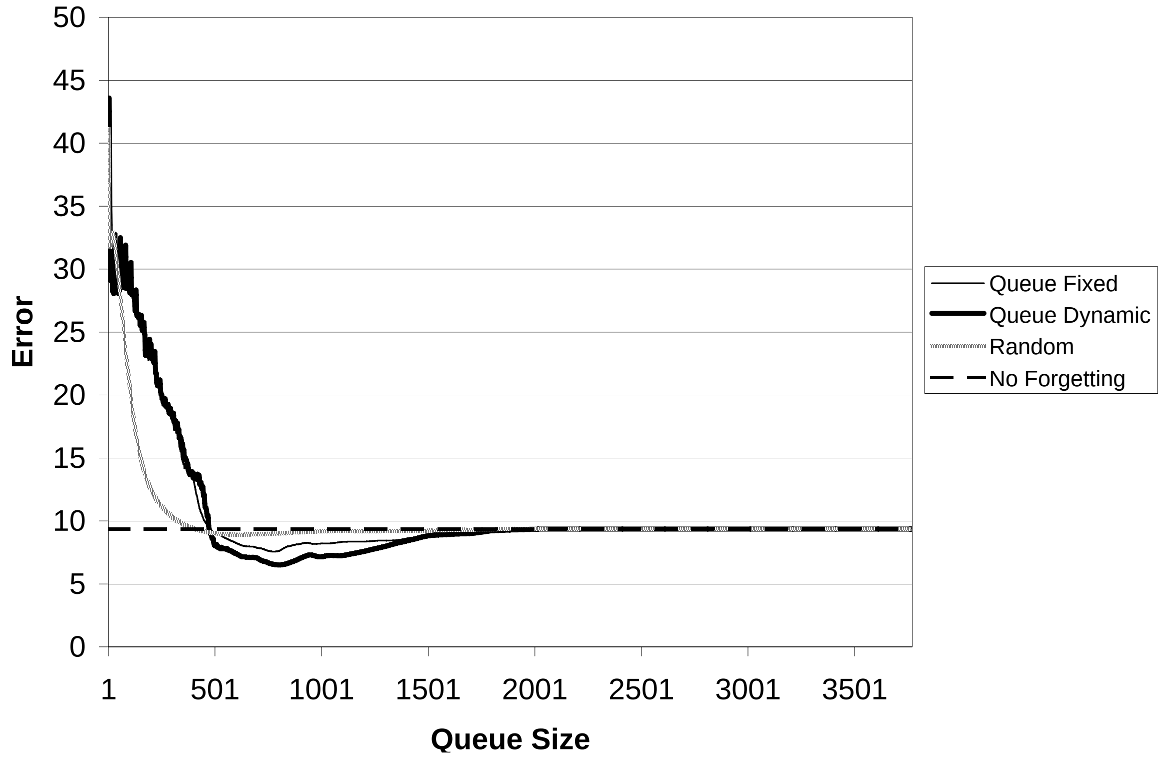
V.1 Optimization

The seven forgetting algorithms described in Chapter IV.1 each possess a diverse collection of properties and parameters. This section presents the two optimization strategies that were employed to determine optimal parameterizations for each of the algorithms. The first optimization approach computed parameters for the Queue Static, Queue Dynamic, and Random forgetting algorithms. ACT-R, SIMPLE, SIMPLE Update, and ActSimple were optimized with a second, more complex process. The optimization used the Nearest Neighbor Interpolation with Perception Count Weighting as the existing robotic algorithm. This interpolation method was selected for its ability to average out the effects of noise.

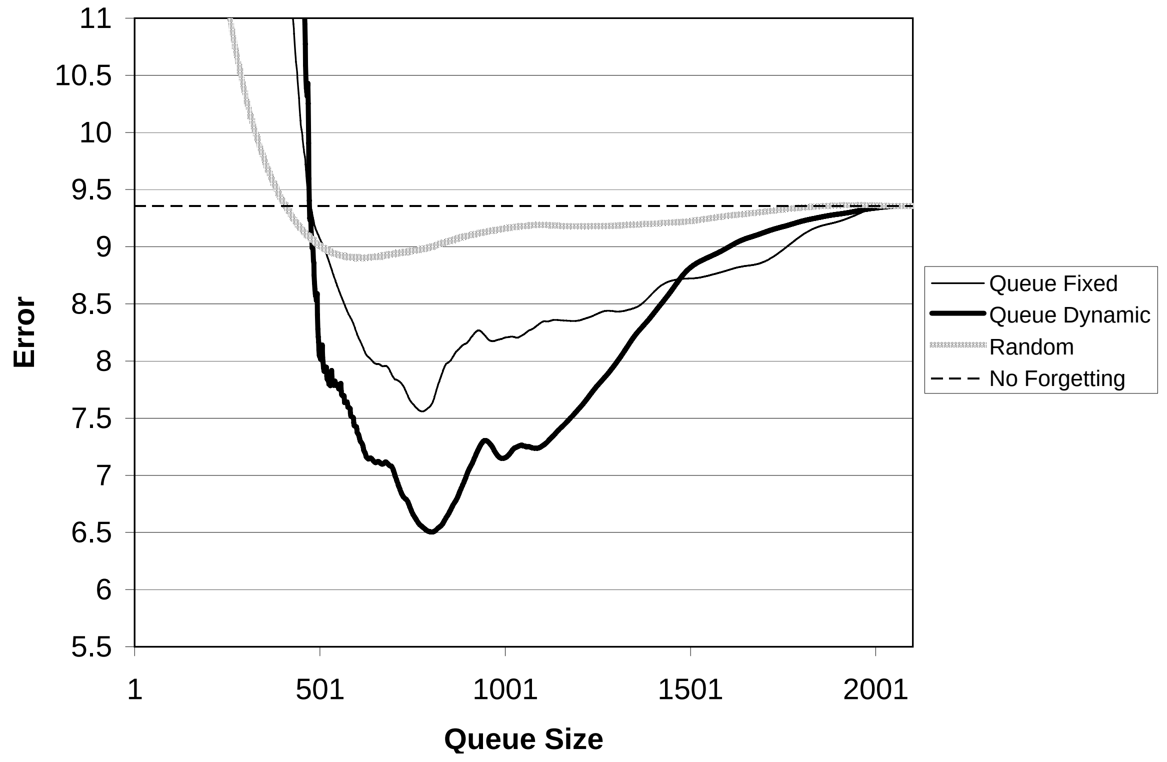
V.1.1 Optimizing the Queue-Based Forgetting Algorithms

The Queue Static, Queue Dynamic, and Random Forgetting algorithms each possesses a single parameter, which is discrete. For these algorithms, an exhaustive search through their respective parameter spaces was completed. The longest path through the simulated environment involved 3768 steps taken by the robot. During the optimization process, the queue size parameter for each of the three forgetting algorithms varied from 1 to 3769. At the largest value, the queue for each algorithm was able to maintain every reading along the path, effectively providing no forgetting. For each queue size, the simulated robot traversed each path 1000 times. Figure V.1a presents the results of the exhaustive search, while Figure V.1b provides a zoomed in perspective of the forgetting algorithms' optimal queue sizes. In both figures, the horizontal dashed line represents the estimation error when no forgetting is performed by the robot.

With small queue sizes, the queue-based forgetting algorithms initially performed poorly (estimation error over 25), but the estimation error quickly decreased as queue size increased (see Figure V.1). Random Forgetting was the first forgetting algorithm to outperform No Forgetting (queue size of 409). As queue size



(a) Full results



(b) Zoomed results

Figure V.1: Queue-based forgetting algorithm optimization results

continued to increase, Queue Static and Queue Dynamic Forgetting began generating less error than Random Forgetting. Performance exceeded No Forgetting levels for queue sizes of 409 - 1868 readings for Random Forgetting, 476 - 2009 readings for Queue Static Forgetting, and 473 - 2125 readings for Queue Dynamic Forgetting. Through the queue size range of 500 to 1400 readings, Queue Dynamic generated error values appreciably better than Queue Static, although both algorithms appeared to follow a similar trend. At larger queue sizes, the performance of the three forgetting algorithms become identical to No Forgetting levels.

V.1.2 Optimizing the Non-Queue-Based Forgetting Algorithms

Ideally, an exhaustive search would have been conducted through the parameter spaces of each remaining forgetting algorithm. However, ACT-R, SIMPLE, SIMPLE Update, and ActSimple each involve a large number of continuous parameters, making this approach infeasible. Instead, a two-stage optimization process was conducted, involving a parameter sweep and a local optimization pattern search. Throughout this optimization process, an assumption was made that the influence of the X and Y axes would be identical in Simple, Simple Update, and ActSimple. The first stage involved sweeping through each algorithm's parameter space by evaluating estimation error at a large number of roughly equi-distant points (5,359,616 for ACT-R, 2,744,560 each for SIMPLE and SIMPLE Update, and 2,621,892 for ActSimple). During this step, the robot traversed each path ten times. Table V.1 lists the number of tested parameterizations for each forgetting algorithm where the average estimation error was below certain threshold limits. The column titled Error Limit provides the threshold values, while the four columns on the right contain the number of parameterizations for each forgetting algorithm that generated estimation errors below the threshold values.

Once the parameter sweep was completed, the Coliny Pattern Search algorithm from the DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) toolkit (DAKOTA, 2010) was used to perform the local optimization. ACT-R, SIMPLE, SIMPLE Update, and ActSimple are all stochastic algorithms and the simulated test environment possesses significant noise, limiting the effectiveness of gradient-based optimization methods. While slower than gradient-based optimization algorithms, the Coliny Pattern Search method maintains greater immunity to noise present in the WiFi readings and estimated signal strength values. During this phase, the robot traversed each path 100 times per test point in a forgetting algorithm's parameter space.

The pattern search was applied to the best 700 parameterizations found during the parameter sweep for each forgetting algorithm. Table V.2 presents the average reduction in estimation error observed for each parameterization that underwent the optimization process. The ACT-R forgetting algorithm experienced a minimal improvement (0.016), while the performance gains for SIMPLE were negligible (0.001). However, ActSimple received a modest improvement (0.250) and SIMPLE Update's performance unexpectedly im-

Table V.1: Parameter sweep counts

Error Limit	ACT-R	SIMPLE	SIMPLE Update	ActSimple
5.8	0	0	0	0
5.9	0	0	0	94
6.0	0	0	0	244
6.1	0	0	0	393
6.2	0	0	0	620
6.3	0	0	0	1,103
6.4	0	0	0	1,979
6.5	541	0	0	3,019
7.0	71,867	0	0	33,963
7.5	129,441	0	0	67,132
8.0	196,401	0	0	93,171
8.5	259,196	0	0	118,156
9.0	328,827	0	0	153,543
9.1	344,216	0	0	159,644
9.3	396,624	0	799	191,870
9.4	1,735,650	917,956	933,614	1,084,886
9.5	1,740,271	938,887	954,086	1,091,645
10.0	1,757,893	985,154	998,773	1,108,362
Points Tested	5,359,616	2,744,560	2,744,560	2,621,892

proved by a large margin (1.646). Despite the significant gains, the initial poor performance of SIMPLE Update prevented the algorithm from exceeding the performance of ActSimple.

Table V.2: Parameterization error improvement

Algorithm	Average	Std.Dev.
ACT-R	0.016	0.017
SIMPLE	0.001	0.001
SIMPLE Update	1.646	1.178
ActSimple	0.250	0.127

V.1.3 Optimization Results

The results from both optimization approaches were combined so that the relative performance of the seven forgetting algorithms could be compared against each other and No Forgetting. The resultant parameterizations and estimation error for each forgetting algorithm and No Forgetting are presented in Table V.3. The column labeled Error presents the estimation error generated by each forgetting algorithm's optimal parameterization. The columns on the right show the individual parameter values for each optimized parameterization. Parameters not included in a forgetting algorithm are indicated with a '—' symbol. No Forgetting does not involve any parameters. All of the forgetting algorithms outperformed No Forgetting, although

SIMPLE's benefits were negligible (an improvement of 0.004). The optimized SIMPLE Update forgetting algorithm generated an average estimation error 3.113 lower than SIMPLE, demonstrating the benefits of updating the perception time associated with a WiFi signal strength reading. The optimized parameterizations for these two algorithms possess large Time-Axis parameter values (0.7035 for SIMPLE and 0.9891 for SIMPLE Update), signifying the importance of time effects.

Table V.3: Best algorithm parameterizations

		Parameters									
Algorithm	Error	Queue Size	β	d	X & Y	Time	Strength	c_3	c_8	ς	τ
No Forgetting	9.357	—	—	—	—	—	—	—	—	—	—
Queue Static	7.559	776	—	—	—	—	—	—	—	—	—
Queue Dynamic	6.504	803	—	—	—	—	—	—	—	—	—
Random	8.892	597	—	—	—	—	—	—	—	—	—
ACT-R	6.451	—	6.1290	0.8257	—	—	—	—	—	104.6835	0.8065
Simple	9.353	—	—	—	0.05553	0.7035	0.1854	2.5000	—	7.7329	-1.2500
Simple Update	6.240	—	—	—	0.004974	0.9891	0.0009642	14.3125	—	5465.8491	0.03125
ActSimple	5.608	—	5.0000	0.9229	0.5000	—	0.0000	2.5000	0.1517	149.2500	0.0000

The ACT-R algorithm's best parameterization generated an estimation error average of 6.451, while ActSimple's best parameterized realized an average value of 5.608. While both of the SIMPLE forgetting algorithms and ACT-R performed worse than ActSimple, these results suggest that the combination of trace-based decay and similarity-based interference can be combined to generate greater performance levels. The optimized parameterization of ActSimple possessed a c_8 value of 0.1517, indicating the importance of similarity-based interference. The X and Y Axis parameters were 0.5000, indicating that only spatial information was used in calculating the similarity-based interference. Strength values of the WiFi readings were not used by the optimized parameterization of ActSimple. The d parameter for ActSimple and ACT-R was 0.9229 and 0.8257 respectively, deviating from the value of 0.5, which is commonly used in many ACT-R based systems.

During the parameter sweep of ACT-R, SIMPLE, SIMPLE Update, and ActSimple's parameter spaces, ten instances of each path were employed to generate average estimation error values and 100 instances were used during the pattern search. Since the forgetting algorithms are stochastic algorithms operating over noisy WiFi readings, the estimation error values are variable. An analysis of this variability was conducted by processing groups of path instances with each forgetting algorithm. Using the optimized parameterizations, each forgetting algorithm processed sets of 10, 100, and 1000 paths instances. The results were collected for 100 trials of each path instance count. The estimation error was then averaged across trials and the results are presented in Table V.4.

While the standard deviation between trials for each forgetting algorithm was reduced as the path instance count increased, the values were all relatively small. These results suggest that the forgetting algorithms are

Table V.4: Variance test

Algorithm	Instances	Average	Std. Dev.	Span
ACT-R	10	6.469	0.027	0.134
	100	6.467	0.009	0.042
	1000	6.469	0.003	0.013
SIMPLE	10	9.358	0.019	0.080
	100	9.358	0.006	0.034
	1000	9.359	0.002	0.008
SIMPLE Update	10	6.247	0.033	0.176
	100	6.253	0.008	0.034
	1000	6.253	0.003	0.014
ActSimple	10	5.629	0.029	0.157
	100	5.631	0.009	0.050
	1000	5.633	0.003	0.014

robust to differences between path instances.

A forgetting algorithm's performance when applied to a particular path may differ from that forgetting algorithm's average performance. Path properties, including basestation configuration changes and length can alter the operation of a forgetting algorithm. The seven forgetting algorithms and No Forgetting were tested with a new set of 100 instances of the twenty paths, described in Chapter IV. The average number of recallable readings generated by each forgetting algorithm or No Forgetting, when applied to each path, is presented in Table V.5. Ranking was performed after rounding to minimize the effects of noise. No Forgetting averaged the largest number of recallable readings (1270.13) although the SIMPLE algorithm generated a nearly identical number of recallable readings (1270.1). The Random forgetting algorithm was limited by its queue size and realized the least number of recallable readings (597). ActSimple produced the second smallest number of recallable readings (636.958). While the SIMPLE Update algorithm averaged 1025.51 recallable readings, ACT-R generated an average of 674.86.

The estimation error generated by each forgetting algorithm or No Forgetting, when applied to each path, is presented in Table V.6. No Forgetting and the SIMPLE algorithm generated identical average estimation error and standard deviation values (9.159 and 6.568, respectively). All other forgetting algorithms generated less overall estimation error. The standard deviation value of 6.568 was also the largest generated by any forgetting algorithm or No Forgetting. ActSimple realized the smallest overall estimation error (5.448), while SIMPLE Update and ACT-R were second and third, respectively (6.058 and 6.276, respectively). Queue Dynamic's fourth smallest estimation error (6.319) was 0.043 larger than ACT-R's estimation error.

No Forgetting generated substantially greater estimation error on the six paths containing a basestation configuration change than the fourteen static paths. Paths 17 and 19 (the two randomly generated dynamic

Table V.5: Individual path average number of recallable readings

	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 1 - 1 Cycle 4 - 8 Basestations	1155.93	776	803	597	635.36	1155.91	734.1	661.28
Path 2 - 1 Cycle 4 Basestations	913.78	776	803	597	631.47	913.74	911.36	688.79
Path 3 - 1 Cycle 4 Basestations	787.8	775.48	787.58	597	583.15	787.79	787.34	671.33
Path 4 - 1 Cycle 4 Basestations	752.33	752.32	752.33	597	558.54	752.3	752.3	663.87
Path 5 - 1 Cycle 8 - 4 Basestations	1155.41	776	803	597	634.7	1155.38	741.15	659.57
Path 6 - 1 Cycle 8 Basestations	973.85	776	803	597	631.47	973.83	915.16	685.64
Path 7 - 1 Cycle 8 Basestations	835.1	776	803	597	588.88	835.09	834.05	671.01
Path 8 - 1 Cycle 8 Basestations	796.6	775.87	794.42	597	565.99	796.58	795.87	667.26
Path 9 - 3 Cycles 4 - 8 Basestations	2048	776	803	597	781.39	2047.94	1170.33	579.68
Path 10 - 3 Cycles 4 Basestations	1484.33	776	803	597	802.86	1484.27	1480.58	701.36
Path 11 - 3 Cycles 4 Basestations	1286.69	776	803	597	732.45	1286.68	1285.93	661.39
Path 12 - 3 Cycles 4 Basestations	1206.36	776	803	597	679.43	1206.3	1201.08	633.74
Path 13 - 3 Cycles 8 - 4 Basestations	2048.21	776	803	597	759.35	2048.15	1111.5	586.78
Path 14 - 3 Cycles 8 Basestations	1687.46	776	803	597	839.68	1687.38	1504.52	683.77
Path 15 - 3 Cycles 8 Basestations	1450.61	776	803	597	767.2	1450.6	1433.73	654.33
Path 16 - 3 Cycles 8 Basestations	1370.07	776	803	597	724.11	1370.03	1338.93	634.38
Path 17 - Random 4 - 8 Basestations	1502.93	776	803	597	691.51	1502.89	775.2	556.41
Path 18 - Random 4 Basestations	1167.63	776	803	597	646.72	1167.6	1041.59	606.96
Path 19 - Random 8 - 4 Basestations	1502.51	776	803	597	646.37	1502.46	740.5	501.12
Path 20 - Random 8 Basestations	1277.1	776	803	597	596.57	1277.07	954.94	570.49
Average	1270.13	774.784	799.266	597	674.86	1270.1	1025.51	636.958
Std	380.540	5.289	11.698	0	83.496	380.526	270.409	53.041
Rank	1	5	4	8	6	2	3	7

paths) resulted in the largest estimation error of the six dynamic paths. Old readings are not filtered by No Forgetting and their influence detrimentally impacts the ability to estimate signal strength. No Forgetting's poor overall estimation error was not simply a result of generating substantial error on dynamic paths. While No Forgetting outperformed ActSimple on the six static one cycle paths (paths 2-4 and 6-8), No Forgetting resulted in greater estimation error with the the six static three cycle paths (paths 10-12 and 14-16).

The statistical significance of the relative estimation error performance results from Table V.6 was tested with two non-parametric Wilcoxon signed rank tests. Non-parametric testing was performed because the true path distribution is not known. Wilcoxon signed rank tests were selected because they can be more powerful than the simpler sign test. Results from the tests are presented in Table V.7. The first test, labeled *Less than No Forgetting*, tested if the forgetting algorithms listed in the left-most column generated less average estimation error than No Forgetting. The second test, labeled *ActSimple less than*, tested if ActSimple resulted is less estimation error than No Forgetting and the other forgetting algorithms. Both tests were singled-sided and used an α of 0.05. Results that were found to be significant are indicated in bold and individual pairings that were not performed are marked with a —— symbol. The tests included the estimation error results from all twenty paths.

ActSimple's reduction of estimation error relative to No Forgetting was found to be significant ($v = 32.0$, $p = 0.0024$), while the reduction of estimation error of the other forgetting algorithms was not found to be significant (column *Less than No Forgetting*). When ActSimple was compared against No Forgetting and the other forgetting algorithms (column *ActSimple less than*), ActSimple's relative estimation error benefits

Table V.6: Individual path estimation error

	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 1 - 1 Cycle 4 - 8 Basestations	17.801	8.987	8.687	14.649	5.635	17.801	8.427	6.024
Path 2 - 1 Cycle 4 Basestations	4.413	5.211	4.553	5.397	4.698	4.413	4.433	4.621
Path 3 - 1 Cycle 4 Basestations	5.004	5.093	5.002	5.625	5.694	5.004	5.01	5.077
Path 4 - 1 Cycle 4 Basestations	5.708	5.708	5.708	6.224	10.944	5.708	5.708	5.746
Path 5 - 1 Cycle 8 - 4 Basestations	18.729	8.736	8.404	15.044	4.897	18.729	5.871	5.296
Path 6 - 1 Cycle 8 Basestations	5.062	6.212	5.335	6.553	5.575	5.062	5.593	5.479
Path 7 - 1 Cycle 8 Basestations	6.098	6.561	6.166	7.135	6.801	6.098	6.103	6.389
Path 8 - 1 Cycle 8 Basestations	6.408	6.559	6.404	7.315	9.055	6.408	6.414	6.557
Path 9 - 3 Cycles 4 - 8 Basestations	17.032	5.972	5.169	8.196	3.834	17.032	5.658	4.618
Path 10 - 3 Cycles 4 Basestations	3.788	4.959	4.128	5.583	1.908	3.789	3.831	2.274
Path 11 - 3 Cycles 4 Basestations	4.404	5.653	3.761	6.103	2.748	4.405	4.412	2.964
Path 12 - 3 Cycles 4 Basestations	5.108	11.604	9.669	7.292	8.907	5.109	5.203	3.715
Path 13 - 3 Cycles 8 - 4 Basestations	18.158	5.271	4.31	7.402	2.837	18.158	4.23	3.502
Path 14 - 3 Cycles 8 Basestations	3.891	6.015	5.128	6.781	2.613	3.892	4.552	3.297
Path 15 - 3 Cycles 8 Basestations	5.005	8.043	5.82	7.648	3.949	5.005	5.015	4.416
Path 16 - 3 Cycles 8 Basestations	5.346	9.434	7.55	8.934	6.765	5.346	5.431	4.947
Path 17 - Random 4 - 8 Basestations	20.971	8.15	7.199	16.624	13.983	20.972	8.756	13.864
Path 18 - Random 4 Basestations	4.762	9.352	5.648	7.172	4.406	4.763	6.145	3.964
Path 19 - Random 8 - 4 Basestations	20.256	7.887	6.673	15.419	12.416	20.258	8.216	10.317
Path 20 - Random 8 Basestations	5.227	12.224	11.069	9.063	7.849	5.227	12.161	5.897
Average	9.159	7.382	6.319	8.708	6.276	9.159	6.058	5.448
Std	6.568	2.148	1.936	3.604	3.355	6.568	1.983	2.616
Rank	7	5	4	6	3	7	2	1

Table V.7: Comparison of estimation error

	Less than No Forgetting			ActSimple less than		
	n	v	p	n	v	p
No Forgetting	——	——	——	20	32.0	**0.0024**
Queue Static	20	105.0	0.5073	20	30.0	**0.0018**
Queue Dynamic	19	81.0	0.2974	20	58.0	**0.0413**
Random	20	108.0	0.5508	20	0.0	<**0.0001**
ACT-R	20	71.0	0.1081	20	71.0	0.1081
SIMPLE	20	154.5	0.9687	20	32.0	**0.0024**
SIMPLE Update	20	105.0	0.5073	20	53.0	**0.0266**
ActSimple	20	32.0	**0.0024**	20	——	——

were found to be significant when compared against No Forgetting ($v = 32.0$, $p = 0.0024$), Queue Static ($v = 30.0$, $p = 0.0018$), Queue Dynamic ($v = 58.0$, $p = 0.0413$), Random ($v = 0.0$, $p < 0.0001$), SIMPLE ($v = 32.0$, $p = 0.0024$), and SIMPLE Update ($v = 53.0$, $p = 0.0266$). However, ActSimple's estimation error improvement over ACT-R was not significant ($v = 71.0$, $p = 0.1081$). These results suggest that for collections of paths similar to the twenty tested paths, ActSimple is an effective approach to reducing average estimation error. While ACT-R's reduction of average estimation error relative to No Forgetting was not found to be significant, ActSimple's estimation error improvement over ACT-R was also not found to be significant. This suggests that the trace-based decay present in ActSimple and ACT-R may play a critical role in the ability to improve performance.

The presence of dynamism can have a large impact on the absolute and relative effectiveness of a forgetting algorithm, and the statistical results presented in Table V.7 may be affected by varying the percentage of paths containing a basestation configuration change. A single-sided Wilcoxon rank sum test was performed to evaluated if No Forgetting and the forgetting algorithms generated less average estimation error on static paths as compared to dynamic paths. This test was employed instead of the Wilcoxon signed rank test because the data was not paired. The test was performed with $\alpha = 0.05$, $n_A = 14$, and $n_B = 6$. Results from this test are presented in Table V.8.

Table V.8: The effects of dynamism on estimation error

	w	p
No Forgetting	0.0	<**0.0001**
Queue Static	37.0	0.3590
Queue Dynamic	31.0	0.1985
Random	5.0	**0.0005**
ACT-R	37.0	0.3590
SIMPLE	0.0	<**0.0001**
SIMPLE Update	25.0	0.0893
ActSimple	26.0	0.1037

The difference in average estimation error between static and dynamic paths was found to be significant for No Forgetting ($w = 0.0$, $p < 0.0001$), Random ($w = 5.0$, $p = 0.0005$), and SIMPLE ($w = 0.0$, $p < 0.0001$). The remaining forgetting algorithms were not found to generate a significant difference in average estimation error. These results suggest that Queue Static, Queue Dynamic, ACT-R, SIMPLE Update, and ActSimple may not be affected by dynamism to the same extent as No Forgetting, Random, and SIMPLE.

Dynamism may affect the relative estimation error generated by No Forgetting and the forgetting algorithms. Four single-sided Wilcoxon signed rank tests were performed to test the differences in estimation error that results from applying No Forgetting and the forgetting algorithms to just static paths and to dynamic paths. Results from these tests are presented in Table V.9. The first test (*Less than No Forgetting - Static*) tests if the forgetting algorithms listed in the left-most column generated less estimation error when only results from the fourteen static paths were considered. The second test (*Less than No Forgetting - Dynamic*) tests if the forgetting algorithms generated less estimation error than No Forgetting when only results from the six dynamic paths were included. The third test (*ActSimple less than - Static*) tests if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms when applied to only the static paths. The fourth test (*ActSimple less than - Dynamic*) tests if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms when applied to only the dynamic paths. The α value was 0.05 for each test and significant results are indicated in bold.

Table V.9: Estimation error performance on static and dynamic paths

	Less than No Forgetting						ActSimple less than					
	Static			Dynamic			Static			Dynamic		
	n	v	p	n	v	p	n	v	p	n	v	p
No Forgetting	——	——	——	——	——	——	14	32.0	0.1083	6	0.0	**0.0156**
Queue Static	14	105.0	1.0000	6	0.0	**0.0156**	14	3.0	**0.0003**	6	9.0	0.4219
Queue Dynamic	13	81.0	0.9960	6	0.0	**0.0156**	14	21.0	**0.0247**	6	11.0	0.5781
Random	14	105.0	1.0000	6	0.0	**0.0156**	14	0.0	**0.0001**	6	0.0	**0.0156**
ACT-R	14	71.0	0.8794	6	0.0	**0.0156**	14	23.0	**0.0338**	6	14.0	0.7812
SIMPLE	14	95.0	0.9980	6	9.5	0.4375	14	32.0	0.1083	6	0.0	**0.0156**
SIMPLE Update	14	105.0	1.0000	6	0.0	**0.0156**	14	18.0	**0.0148**	6	10.0	0.5000
ActSimple	14	32.0	0.1083	6	0.0	**0.0156**	——	——	——	——	——	——

None of the results from testing if the forgetting algorithms generated less estimation error than No Forgetting when applied to only the static paths (column *Less than No Forgetting - Static*) were found to be significant. However, when the forgetting algorithms were tested to determine if they resulted in less estimation error than No Forgetting when processing dynamic paths (column *Less than No Forgetting - Dynamic*), all of the results were significant ($v = 0.0$, $p = 0.0156$), except for SIMPLE ($v = 9.5$, $p = 0.4375$). All of the forgetting algorithms, except for SIMPLE, generated identical metric and p values because they resulted in less estimation error than No Forgetting when applied to each of the six dynamic paths. These results suggest that Human-Inspired Forgetting may be an effective approach to improving system accuracy in dynamic domains. While the results from comparing the estimation error from ActSimple to the estimation error from No Forgetting when applied to the static paths was not found to be significant, ActSimple's average estimation error on the static paths ($\mu = 4.6674$, $\sigma = 1.303$) was smaller than the average estimation error resulting from No Forgetting ($\mu = 5.0161$, $\sigma = 0.7506$). These results suggest that in addition to the benefits provided by applying Human-Inspired Forgetting to dynamic paths, ActSimple may result in equal or even improved accuracy when employed under some static conditions.

The third Wilcoxon signed rank test in Table V.9 evaluated if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms when only results from the static paths were included (column *ActSimple less than - Static*). The results were significant for Queue Static ($v = 3.0$, $p = 0.0003$), Queue Dynamic ($v = 21.0$, $p = 0.0247$), Random ($v = 0.0$, $p = 0.0001$), ACT-R ($v = 23.0$, $p = 0.0338$), and SIMPLE Update ($v = 18.0$, $p = 0.0148$). The results for No Forgetting and SIMPLE were identical and not significant. The fourth Wilcoxon signed rank test in Table V.9 tested if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms when only results from the dynamic paths were used. The results were significant for No Forgetting ($v = 0.0$, $p = 0.0156$), Random ($v = 0.0$,

$p = 0.0156$), and SIMPLE ($v = 0.0$, $p = 0.0156$). Results for the remaining forgetting algorithms were not significant. These results suggest that Human-Inspired Forgetting may not improve accuracy across all possible conditions. Instead, forgetting algorithms and their parameterizations may need to be optimized for specific domain properties and conditions. ActSimple was able to generate less estimation error than the other forgetting algorithms (excluding SIMPLE) when processing static paths and also resulted in less estimation error than No Forgetting and SIMPLE when applied to dynamic paths. ActSimple may not always result in the least amount of estimation error for every specific tested condition, but these results suggest ActSimple may be capable of improving average system accuracy. The ratio of static to dynamic paths used during the optimization process may have influenced the relative performance of ActSimple and the other forgetting algorithms when applied to static and dynamic paths.

V.1.4 Summary

The optimization process and results for ActSimple and the six other forgetting algorithms have been provided. Two separate optimization procedures were employed, one for the queue-based forgetting algorithms and another for the remaining four algorithms. During optimization, the forgetting algorithms were optimized with WiFi signal strength estimation results from a simulated robot traversing the twenty paths described in Chapter IV.4. Only one metric was used during optimization, the average absolute signal strength estimation error generated by the robot when it employed the forgetting algorithms. After the robot had finished traversing each path, the robot estimated the signal strength at each location within the environment and the resultant error at each location was recorded. The metric was computed by averaging the estimation error for 100 instances of each of the twenty paths.

The forgetting algorithms and No Forgetting were tested for twenty paths after optimization. ActSimple was found to generate the smallest amount of estimation error, while SIMPLE and No Forgetting tied for the greatest amount of estimation error (Table V.6). These results suggest in addition to Human-Inspired Forgetting having the potential to improve performance relative to No Forgetting, ActSimple may be a highly effective implementation. However, while ActSimple outperformed No Forgetting and the other forgetting algorithms, ActSimple possesses the largest number of parameters and can be considered the most complex of the algorithms.

The effects of algorithm complexity are shown in Figure V.2, which plots an algorithm's number of parameters versus its resultant average absolute estimation error. Path dynamism was demonstrated to have an effect on the estimation error generated by forgetting algorithms and No Forgetting. Each forgetting algorithm (and No Forgetting) in Figure V.2 is indicated with a unique symbol, while the set of paths used to create the average estimation error is represented by color. The results of averaging estimation error from all

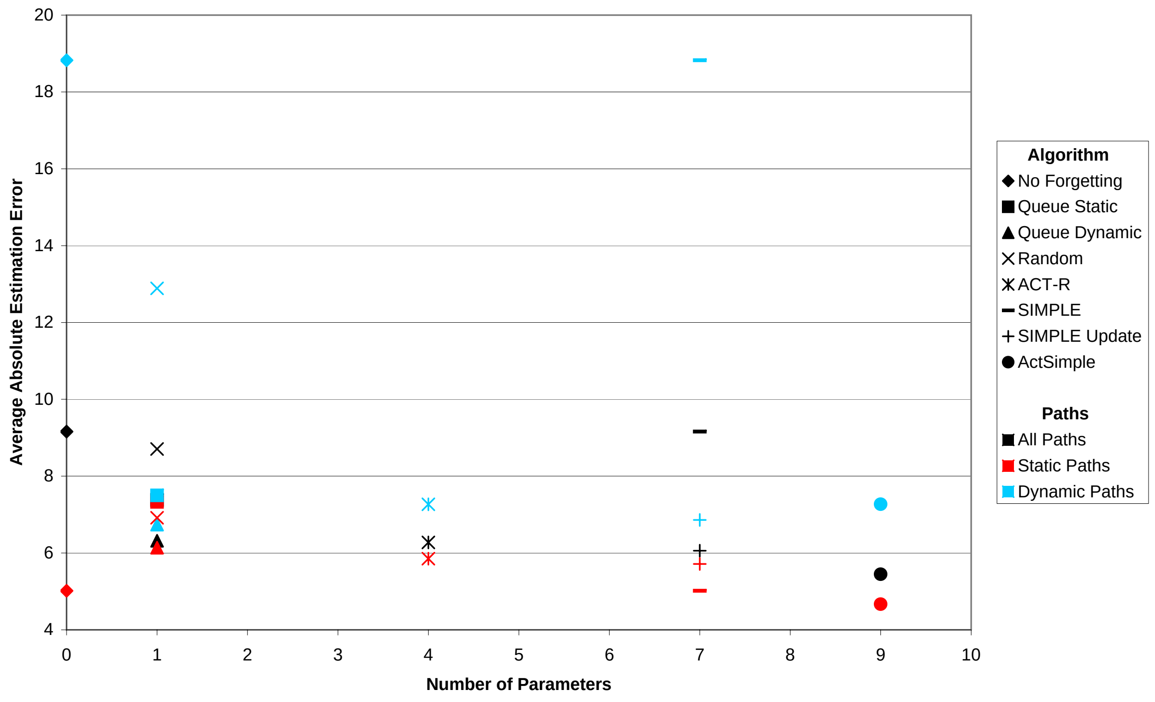


Figure V.2: Number of parameters vs. average absolute estimation error

twenty paths is shown in black, while results generated solely from the fourteen static paths are shown in red. Blue indicates results involving only the six dynamic paths.

Excluding SIMPLE, Figure V.2 shows an inverse relationship between complexity and estimation error when all twenty paths are included (Black). SIMPLE and No Forgetting generated an identical amount of estimation error, suggesting parameter count is not the sole factor influencing performance. The relative performance of the forgetting algorithms and No Forgetting was more variable when the paths were partitioned based on dynamism. While ActSimple generated the smallest amount of estimation error on static paths (Red), No Forgetting resulted in the second smallest amount. When applied to only dynamic paths (Blue), ActSimple generated the fourth smallest amount of estimation error, while No Forgetting resulted in the seventh smallest amount of estimation error. SIMPLE Update generated the second smallest amount of estimation error on dynamic paths, but only resulted in the fourth smallest amount of estimation error on static paths. The optimization process was performed for the entire set of twenty paths and the ratio of static to dynamic paths may have influenced the resultant parameterizations.

In addition to the average estimation error metric (Table V.6), a second metric was recorded, the average number of recallable readings (Table V.5). The average number of recallable readings metric shows the average number of WiFi signal strength readings presented to the existing robotic algorithm (Nearest Neighbor Interpolation with Perception Count Weighting). While No Forgetting resulted in the largest number of re-

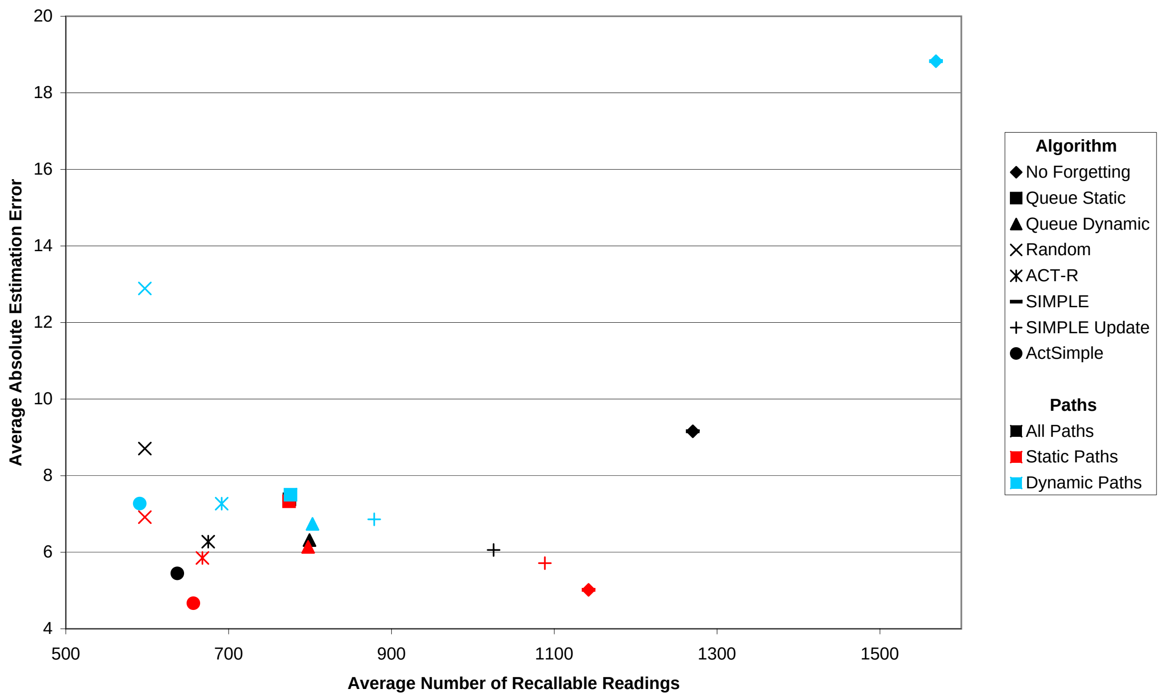


Figure V.3: Average number of recallable readings vs. average absolute estimation error

callable readings, ActSimple filtered the second largest number of readings. Random forgetting averaged the smallest number of recallable readings, suggesting a forgetting algorithm's operation and parameterization have a large impact on the amount of filtering that is performed.

Forgetting algorithm selectivity has a greater influence on performance than just the average number of recallable readings. The relationship between the average number of recallable readings and average absolute estimation error for each forgetting algorithm and No Forgetting is presented in Figure V.3. This figure uses the same key as employed in Figure V.2. When all twenty paths are included, ActSimple filtered the second largest number of readings, but still managed to produce the lowest amount of estimation error. No Forgetting, which filtered the fewest readings, tied with SIMPLE for the largest amount of estimation error. Random forgetting filtered the most readings, but only generated the sixth best estimation error. These results suggest that simply using more data to solve a problem may not always be the optimal solution.

When the results from static and dynamic paths were separated, a performance trend appears for No Forgetting and the forgetting algorithms. Applying No Forgetting, SIMPLE, Queue Static, Queue Dynamic, and ACT-R to only the static paths resulted in fewer recallable readings and less estimation error than when the forgetting algorithms and No Forgetting were applied to the dynamic paths. These forgetting algorithms and No Forgetting performed better with the static paths. The Random forgetting algorithm averaged the same number of recallable readings under all conditions, but still performed better on static paths. ActSimple and SIMPLE Update resulted in a different trend. While both algorithms generated less estimation error when only processing static paths, they also resulted in larger numbers of recallable readings. In situations where the average number of recallable readings can have a noticeable impact on efficiency and performance, determining if these two algorithms perform better with static or dynamic paths is more challenging.

V.2 Alternate Existing Robotic Algorithms

Robots operating within complex and dynamic domains require the ability to adapt to changing task conditions as well as environment fluctuations. In Section V.1, the seven forgetting algorithms were optimized while being applied to the existing robotic algorithm of Nearest Neighbor Interpolation with Perception Count Weighting. This section explores the effects on forgetting algorithm performance when changes occur to the existing robotic algorithm. Results from this experiment reveal if the forgetting algorithms maintain their performance levels, realize graceful degradation, or fail completely.

This experiment tested all four interpolation methods described in Chapter IV.3 (Nearest Neighbor Interpolation with Perception Count Weighting, Nearest Neighbor Interpolation with Equal Weighting, Shepard's Interpolation, and Microsphere Interpolation) with the optimized parameterizations discovered in Chapter V.1. The seven forgetting algorithms and No Forgetting were individually combined with the four interpolation methods and evaluated with 1000 instances each of the twenty paths described in Chapter IV. These path instances were the same as those used in optimizing the queue forgetting methods (Section V.1).

Estimation error generated from combining each forgetting algorithm or No Forgetting with the four interpolations methods is presented in Table V.10. The best performing forgetting algorithms were able to minimize the resultant estimation error for each interpolation method. In the table, the right most two columns present the mean and standard deviation across interpolation method. Error values were smallest when the Nearest Neighbor with Perception Count Weighting method was used, except for Random forgetting where Microsphere Interpolation generated 0.006 less error. The Shepard's Interpolation method consistently produced the largest estimation error and the Nearest Neighbor Interpolation with Equal Weighting method always underperformed the Nearest Neighbor Interpolation with Perception Count Weighting method. The SIMPLE algorithm and No Forgetting generated nearly identical error values. While SIMPLE outperformed No Forgetting by 0.004 when Nearest Neighbor Interpolation with Perception Count Weighting was used, the error values for the other interpolation methods were equivalent.

The rank orderings of the estimation error values were calculated and are presented in Table V.11. The ranking was performed after rounding to minimize the effects of noise. The ActSimple algorithm generated the smallest estimation error for each of the four interpolation methods. The SIMPLE Update algorithm generated the second best error with Nearest Neighbor Interpolation with Perception Count Weighting, but

Table V.10: New interpolation methods

	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	Simple	Simple Update	ActSimple	Mean	Std
NN - Perception Count	9.357	7.559	6.504	8.892	6.451	9.353	6.251	5.608	7.497	1.515
NN - Equal	10.718	7.803	7.015	9.182	6.9	10.718	7.591	5.911	8.23	1.793
Shepard's	11.485	9.169	7.941	11.98	7.836	11.485	8.586	7.263	9.468	1.896
Microsphere	10.609	7.619	6.842	8.886	6.738	10.609	7.414	5.691	8.051	1.818
Average	10.542	8.037	7.076	9.735	6.981	10.542	7.46	6.118	8.311	1.727
Std	0.881	0.761	0.615	1.503	0.6	0.883	0.957	0.774	0.872	0.285

produced the fourth smallest error with the other interpolation methods. While the ACT-R algorithm generated the third best error with Nearest Neighbor Interpolation with Perception Count Weighting, the forgetting algorithm achieved the second best error with the remaining interpolation methods. With Nearest Neighbor Interpolation with Perception Count Weighting, the Queue Dynamic algorithm generated the fourth best estimation error and improved to third for the remaining interpolation methods. The Queue Static algorithm consistently achieved the fifth best error. The Random and SIMPLE algorithms, along with No Forgetting, never realized a ranking better than sixth.

Table V.11: New interpolation method rankings

	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	Simple	Simple Update	ActSimple	Mean	Std
NN - Perception Count	8	5	4	6	3	7	2	1	4.5	2.449
NN - Equal	7	5	3	6	2	7	4	1	4.375	2.264
Shepard's	6	5	3	8	2	6	4	1	4.375	2.326
Microsphere	7	5	3	6	2	7	4	1	4.375	2.264
Average	7	5	3.25	6.5	2.25	6.75	3.5	1	4.406	2.248
Std	0.816	0	0.5	1	0.5	0.5	1	0	0.54	0.394

The estimation error differences between Nearest Neighbor Interpolation with Perception Count Weighting and the other interpolation methods are shown in Table V.12. Positive deviation values reflect a decrease in performance. Except for the marginal performance gain (-0.006) for the Random forgetting algorithm and Microsphere Interpolation, the forgetting algorithms and No Forgetting each experienced a decrease in performance when an interpolation method other than Nearest Neighbor Interpolation with Perception Count Weighting was employed. The performance decrements were not consistent and ranged from 0.060 to 3.088. The ActSimple algorithm generated the second smallest average performance decrement, trailing only Queue Static forgetting. No Forgetting generated increasing estimation error for each of the three alternate interpolation methods. These increases in error suggest the alternate interpolation methods pose a greater challenge to the forgetting algorithms. Maintaining the estimation error values that were generated with Nearest Neighbor Interpolation with Perception Count Weighting requires the forgetting algorithms to increase the amount of error reduction afforded by filtering. The estimation error increases for the Queue Static, Queue Dynamic, ACT-R, and ActSimple algorithms were all smaller than the respective changes with No Forgetting.

These four forgetting algorithms maintained their benefits to WiFi signal strength estimation, improving their relative benefit when applied to the three new and more challenging existing robotic algorithms. These estimation error trends suggest the algorithms are robust to changes in task. Algorithm performance will degrade gracefully if a robot's task changes slightly or is poorly modeled.

Table V.12: New interpolation method performance deviations

	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	Simple	Simple Update	ActSimple	Mean	Std
NN - Equal	1.361	0.243	0.511	0.29	0.449	1.365	1.34	0.303	0.733	0.523
Shepard's	2.128	1.609	1.437	3.088	1.386	2.132	2.335	1.655	1.971	0.572
Microsphere	1.252	0.06	0.338	-0.006	0.287	1.256	1.163	0.083	0.554	0.567
Average	1.581	0.638	0.762	1.124	0.707	1.585	1.613	0.681	1.086	0.445
Std	0.477	0.847	0.591	1.707	0.593	0.477	0.632	0.851	0.772	0.404

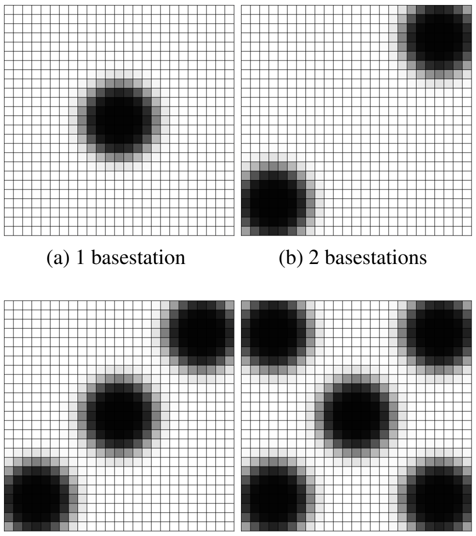
V.3 Alternate Basestation Configurations and Paths

The seven forgetting algorithms were optimized and compared against No Forgetting while filtering WiFi signal strength readings. During optimization, the Nearest Neighbor Interpolation with Perception Count Weighting was employed as the existing robotic algorithm. Section V.2 demonstrated how changes to the existing robotic algorithm (interpolation method) can affect the performance of a forgetting algorithm. Performance can also be influenced by changes to the environment. This section describes an experiment where the forgetting algorithms, along with No Forgetting, were applied to a set of 44 new paths (labeled path 21 - 64), comprised of new path trajectories, basestation configurations, and number of basestation configuration changes that occurred during a path.

V.3.1 Description of Paths 21 - 64

The paths used during the optimization process employed two basestation configurations, one with four basestations and another with eight (Figure IV.1). During the experiment presented in this section, four new basestation configurations were constructed. These new configurations consisted of: a single basestation configuration, a two basestation configuration, a three basestation configuration, and a five basestation configuration. The signal strength available at each location in the environment for each basestation configuration is presented in Figure V.4.

The 44 new paths were created with three different types of path trajectories. Twenty of the paths (Path 21 - 40) incorporated an exhaustive pattern through the environment (Figure V.5a). This trajectory is identical to the trajectory of Paths 1, 2, 5, 6, 9, 10, 13, and 14. The trajectory starts at the center of the environment and then travels to the lower left corner. From this point, the trajectory exhaustively moves to the upper right corner of the environment. The trajectory then returns to the lower left corner. Paths using this trajectory



(a) 1 basestation (b) 2 basestations

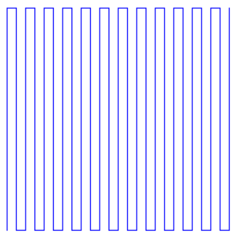
(c) 3 basestations (d) 5 basestations

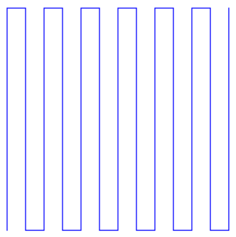
Figure V.4: New basestation configurations

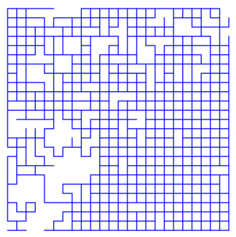
Table V.13: Basestation configurations for paths 21 - 64

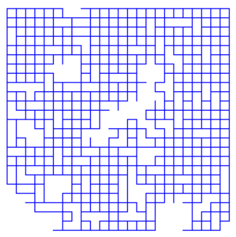
Path	Cfg.	Path	Cfg.	Path	Cfg.	Path	Cfg.
Path 21	1	Path 32	5	Path 43	3	Path 54	2 - 1
Path 22	2	Path 33	1 - 2	Path 44	5	Path 55	3 - 5
Path 23	3	Path 34	2 - 1	Path 45	1 - 2	Path 56	5 - 3
Path 24	5	Path 35	3 - 5	Path 46	2 - 1	Path 57	1
Path 25	1 - 2	Path 36	5 - 3	Path 47	3 - 5	Path 58	2
Path 26	2 - 1	Path 37	1 - 2 - 1	Path 48	5 - 3	Path 59	3
Path 27	3 - 5	Path 38	2 - 1 - 2	Path 49	1	Path 60	5
Path 28	5 - 3	Path 39	1 - 2 - 3	Path 50	2	Path 61	1 - 2
Path 29	1	Path 40	3 - 2 - 1	Path 51	3	Path 62	2 - 1
Path 30	2	Path 41	1	Path 52	5	Path 63	3 - 5
Path 31	3	Path 42	2	Path 53	1 - 2	Path 64	5 - 3

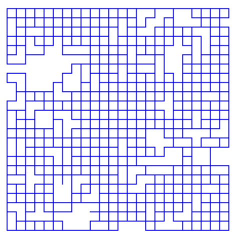
made either one or three round trips from the lower left corner to the upper right and back. The second trajectory (Figure V.5b) is similar to the first, except every other column in the environment is skipped. This trajectory was used by Paths 41 - 56. The remaining trajectories (Figures V.5c - V.5j) were each randomly generated. Each of these trajectories were used by only one path (Paths 57 - 64). Paths 21 - 28 and 41 - 48 were 1272 steps long while paths 29 - 40 and 49 - 56 were 3768 steps long. The randomly generated paths (Paths 57 - 64) were all 3000 steps in length. Twenty of the paths possessed one basestation configuration change. This change occurred at roughly the midpoint of the paths. Four paths incorporated two basestation configuration changes. These four paths changed configurations at approximately the one third and two thirds points along the path. Table V.13 presents the basestation configurations employed by each path.

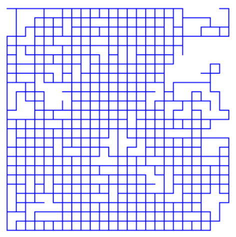

(a) Exhaustive path

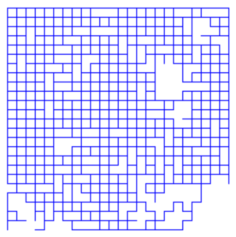

(b) Sparse path

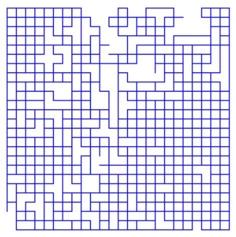

(c) Path 57

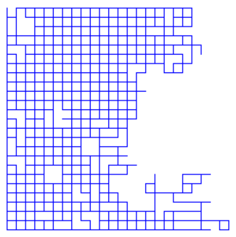

(d) path 58


(e) Path 59


(f) Path 60


(g) Path 61


(h) Path 62


(i) Path 63

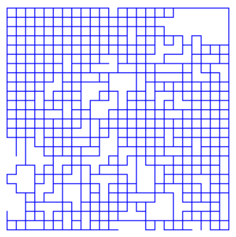

(j) Path 64

Figure V.5: New path trajectories

V.3.2 Forgetting Performance with Paths 21 - 64

The seven forgetting algorithms and No Forgetting were applied to 100 instances of each new path. Testing conditions were identical to those used during optimization except for the actual paths. The Nearest Neighbor Interpolation with Perception Count Weighting was used as the existing robotic algorithm.

The resultant individual average number of recallable readings are presented in Table V.14. Ranking values were computed after rounding to minimize the effects of noise. The rank ordering when only paths 21 - 64 were included in the results (Table V.14) was identical to the rank ordering when only paths 1 - 20 were tested (Table V.5). No Forgetting averaged the largest number of recallable readings (1090.794), with SIMPLE averaging nearly the same amount (1090.759). ActSimple averaged the second least number of recallable readings (600.877) and the Random forgetting algorithm filtered the most (583.108). The Random and ActSimple algorithms also generated the smallest standard deviations (32.05 and 71.07, respectively).

The resultant individual estimation error results are presented in Table V.15. The rank ordering of the forgetting algorithms and No Forgetting differed from the rank orderings when the forgetting algorithms and No Forgetting were applied to paths 1 - 20 (Table V.6), unlike the average number of recallable reading count rankings. The ActSimple algorithm maintained the smallest estimation error (5.144), but SIMPLE Update moved from second place to sixth. ActSimple's estimation error was 5.448 when applied to paths 1 - 20 and decreased to 5.144 when applied to paths 21 - 64. The SIMPLE Update algorithm's estimation error increased from 6.058 to 8.708. The ACT-R algorithm generated a third place estimation error (6.276) with paths 1 - 20, but improved to second place by producing an estimation error of 5.277 with paths 21 - 64. Both Queue Static and Queue Dynamic improved their rankings by one despite generating larger estimation error for paths 21 - 64. Queue Static produced an estimation error of 7.382 for paths 1 - 20 and 7.571 for paths 21 - 64. The Queue Dynamic algorithm generated an estimation error of 6.319 for paths 1- 20 and 6.675 for paths 21 - 64. No Forgetting and the SIMPLE algorithm tied for the largest estimation error (9.058) in a similar fashion to their performance with paths 1 - 20.

The statistical significance of the relative estimation error performance results from Table V.15 was tested with two Wilcoxon signed rank tests and are presented in Table V.16. These tests were performed in a similar fashion to those presented in Chapter V.1.3. The first test, labeled *Less than No Forgetting*, tested if the forgetting algorithms listed in the left-most column generated less average estimation error than No Forgetting. The second test, labeled *ActSimple less than*, tested if ActSimple resulted is less estimation error than No Forgetting and the other forgetting algorithms. Both tests were singled-sided and used an α of 0.05. Results that were found to be significant are indicated in bold and individual pairings that were not performed are marked with a —— symbol. The tests included the estimation error results from all 44 paths.

Table V.14: Average number of recallable readings for paths 21 - 64

	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 21 - 1 Cycle 1 Basestations	867.76	776	803	597	629.62	867.7	718.59	684.02
Path 22 - 1 Cycle 2 Basestations	877.33	776	803	597	630.58	877.32	781.46	687.76
Path 23 - 1 Cycle 3 Basestations	907.48	776	803	597	631.11	907.41	832.66	684.53
Path 24 - 1 Cycle 5 Basestations	944.44	776	803	597	631.41	944.44	943.33	686.33
Path 25 - 1 Cycle 1 - 2 Basestations	1017.46	776	803	597	641.24	1017.43	712.73	675.23
Path 26 - 1 Cycle 2 - 1 Basestations	1019.39	776	803	597	639.84	1019.35	698.69	675.19
Path 27 - 1 Cycle 3 - 5 Basestations	1013.18	776	803	597	631.79	1013.13	864.31	675.33
Path 28 - 1 Cycle 5 - 3 Basestations	1015.34	776	803	597	632.01	1015.3	904.21	674.49
Path 29 - 3 Cycles 1 Basestations	1322.37	776	803	597	775.26	1322.34	1171.52	725.32
Path 30 - 3 Cycles 2 Basestations	1358.58	776	803	597	782.94	1358.49	1262.24	717.62
Path 31 - 3 Cycles 3 Basestations	1455.92	776	803	597	800.62	1455.88	1379.26	711.99
Path 32 - 3 Cycles 5 Basestations	1588.12	776	803	597	825.91	1588.02	1577.04	696.9
Path 33 - 3 Cycles 1 - 2 Basestations	1619.48	776	803	597	761.82	1619.36	1080.82	651.95
Path 34 - 3 Cycles 2 - 1 Basestations	1617.03	776	803	597	758.02	1616.97	1035.48	661.49
Path 35 - 3 Cycles 3 - 5 Basestations	1689.26	776	803	597	806.93	1689.19	1388.25	656.67
Path 36 - 3 Cycles 5 - 3 Basestations	1686.57	776	803	597	787.37	1686.52	1240.63	665.66
Path 37 - 3 Cycles 1 - 2 - 1 Basestations	1594.34	776	803	597	754.5	1594.34	1121.57	664.7
Path 38 - 3 Cycles 2 - 1 - 2 Basestations	1611.8	776	803	597	759.65	1611.74	1133.04	655.19
Path 39 - 3 Cycles 1 - 2 - 3 Basestations	1644.36	776	803	597	781.42	1644.32	1146.95	662.66
Path 40 - 3 Cycles 3 - 2 - 1 Basestations	1647.4	776	803	597	804.7	1647.34	1148.84	658.19
Path 41 - 1 Cycle 1 Basestations	478.14	478.14	478.14	478.14	449.12	478.12	478.14	477.75
Path 42 - 1 Cycle 2 Basestations	483.88	483.88	483.88	483.88	451.04	483.88	483.88	483.24
Path 43 - 1 Cycle 3 Basestations	499.63	499.63	499.63	499.63	466.54	499.61	499.63	498.68
Path 44 - 1 Cycle 5 Basestations	521.68	521.68	521.68	521.68	483.87	521.68	521.68	520.74
Path 45 - 1 Cycle 1 - 2 Basestations	558.72	558.72	558.72	558.72	519.75	558.69	558.72	554.78
Path 46 - 1 Cycle 2 - 1 Basestations	558.72	558.72	558.72	558.72	519.22	558.69	558.72	554.71
Path 47 - 1 Cycle 3 - 5 Basestations	560.23	560.23	560.23	560.23	514.55	560.23	560.23	559.3
Path 48 - 1 Cycle 5 - 3 Basestations	559.31	559.31	559.31	559.31	514.47	559.3	559.31	558.29
Path 49 - 3 Cycle 1 Basestations	718.89	718.89	718.89	597	490.36	718.87	718.89	514.79
Path 50 - 3 Cycle 2 Basestations	745.78	745.77	745.78	597	502.4	745.74	745.78	521.64
Path 51 - 3 Cycle 3 Basestations	795.85	775.6	792.73	597	525.56	795.84	795.85	535.33
Path 52 - 3 Cycle 5 Basestations	872.79	776	803	597	559.69	872.77	872.79	555.85
Path 53 - 3 Cycle 1 - 2 Basestations	877.86	776	803	597	570.29	877.83	858.72	544.54
Path 54 - 3 Cycle 2 - 1 Basestations	878.81	776	803	597	559.22	878.79	844.19	532.83
Path 55 - 3 Cycle 3 - 5 Basestations	927.95	776	803	597	604.4	927.94	922.3	577.67
Path 56 - 3 Cycle 5 - 3 Basestations	925.63	776	803	597	567.56	925.63	925.63	542.57
Path 57 - Random 1 Basestations	1106.44	776	803	597	602.72	1106.39	905.6	629.61
Path 58 - Random 2 Basestations	1148.29	776	803	597	622.52	1148.25	962.87	600.2
Path 59 - Random 3 Basestations	1234.58	776	803	597	675.58	1234.57	1070.24	636.7
Path 60 - Random 5 Basestations	1287.83	776	803	597	638.92	1287.79	987.12	569.79
Path 61 - Random 1 - 2 Basestations	1290.66	776	803	597	662.71	1290.61	739.02	584.82
Path 62 - Random 2 - 1 Basestations	1193.83	776	803	597	578.01	1193.79	696.49	564.84
Path 63 - Random-3 - 5 Basestations	1214.35	776	803	597	620.73	1214.34	910.16	548.07
Path 64 - Random 5 - 3 Basestations	1291.32	776	803	597	585.67	1291.28	854.76	539.75
Average	1090.794	724.114	744.118	583.108	630.723	1090.759	897.408	600.877
Std	398.434	101.215	111.859	32.05	116.051	398.417	272.904	71.07
Rank	1	5	4	8	6	2	3	7

Table V.15: Average estimation error for paths 21 - 64

	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 21 - 1 Cycle 1 Basestations	4.063	4.539	4.061	4.606	4.106	4.063	6.563	4.13
Path 22 - 1 Cycle 2 Basestations	4.13	4.627	4.148	4.759	4.229	4.13	5.766	4.181
Path 23 - 1 Cycle 3 Basestations	4.313	5.1	4.436	5.189	4.536	4.313	5.566	4.518
Path 24 - 1 Cycle 5 Basestations	4.573	5.525	4.741	5.713	4.925	4.573	4.578	4.841
Path 25 - 1 Cycle 1 - 2 Basestations	14.371	10.038	8.689	11.895	5.384	14.371	12.255	5.5
Path 26 - 1 Cycle 2 - 1 Basestations	14.529	10.057	8.694	11.982	5.242	14.531	15.777	5.397
Path 27 - 1 Cycle 3 - 5 Basestations	10.727	8.629	7.886	10.097	5.01	10.728	6.603	5.691
Path 28 - 1 Cycle 5 - 3 Basestations	11.141	8.596	7.843	10.184	4.564	11.141	12.931	5.309
Path 29 - 3 Cycles 1 Basestations	3.798	5.851	3.57	5.059	1.443	3.799	6.711	1.765
Path 30 - 3 Cycles 2 Basestations	3.827	4.981	3.756	5.16	1.599	3.827	5.768	1.859
Path 31 - 3 Cycles 3 Basestations	3.82	4.939	4.107	5.516	1.847	3.82	5.277	2.28
Path 32 - 3 Cycles 5 Basestations	3.816	5.507	4.6	6.104	2.238	3.817	3.799	2.809
Path 33 - 3 Cycles 1 - 2 Basestations	13.999	4.877	3.851	6.828	2.763	14	7.84	2.676
Path 34 - 3 Cycles 2 - 1 Basestations	14.119	4.666	3.649	6.71	2.454	14.12	10.439	2.282
Path 35 - 3 Cycles 3 - 5 Basestations	10.023	5.406	4.577	6.903	2.639	10.023	4.644	3.288
Path 36 - 3 Cycles 5 - 3 Basestations	10.643	4.912	4.095	6.602	2.089	10.643	6.445	2.484
Path 37 - 3 Cycles 1 - 2 - 1 Basestations	10.372	4.421	3.838	7.682	1.919	10.372	9.958	2.245
Path 38 - 3 Cycles 2 - 1 - 2 Basestations	10.253	4.599	3.992	7.682	2.127	10.254	7.262	2.687
Path 39 - 3 Cycles 1 - 2 - 3 Basestations	10.047	4.986	4.24	6.78	2.465	10.047	5.903	2.847
Path 40 - 3 Cycles 3 - 2 - 1 Basestations	15.074	4.486	3.889	8.122	6.468	15.075	14.761	2.645
Path 41 - 1 Cycle 1 Basestations	4.335	4.335	4.335	4.335	4.267	4.335	4.335	4.335
Path 42 - 1 Cycle 2 Basestations	4.261	4.261	4.261	4.261	4.266	4.261	4.261	4.261
Path 43 - 1 Cycle 3 Basestations	4.76	4.76	4.76	4.76	4.779	4.76	4.76	4.764
Path 44 - 1 Cycle 5 Basestations	5.39	5.39	5.39	5.39	5.431	5.39	5.39	5.391
Path 45 - 1 Cycle 1 - 2 Basestations	15.135	15.135	15.135	15.135	13.635	15.133	15.135	15.018
Path 46 - 1 Cycle 2 - 1 Basestations	15.188	15.188	15.188	15.188	13.502	15.187	15.188	15.061
Path 47 - 1 Cycle 3 - 5 Basestations	11.306	11.306	11.306	11.306	10.146	11.306	11.306	11.309
Path 48 - 1 Cycle 5 - 3 Basestations	11.631	11.631	11.631	11.631	10.324	11.632	11.631	11.635
Path 49 - 3 Cycles 1 Basestations	4.145	4.145	4.145	4.883	2.35	4.144	4.145	2.644
Path 50 - 3 Cycles 2 Basestations	4.069	4.069	4.069	4.98	2.43	4.07	4.069	2.681
Path 51 - 3 Cycles 3 Basestations	4.453	4.665	4.454	5.574	2.961	4.453	4.453	3.179
Path 52 - 3 Cycles 5 Basestations	4.808	6.182	4.625	6.329	3.596	4.808	4.808	3.717
Path 53 - 3 Cycles 1 - 2 Basestations	14.697	12.842	13.343	12.509	10.299	14.697	13.836	8.429
Path 54 - 3 Cycles 2 - 1 Basestations	14.843	13.241	13.401	12.724	9.435	14.844	12.917	7.601
Path 55 - 3 Cycles 3 - 5 Basestations	10.853	10.548	10.049	10.234	8.927	10.854	10.43	8.052
Path 56 - 3 Cycles 5 - 3 Basestations	11.237	11.25	10.057	10.167	6.946	11.237	11.237	5.632
Path 57 - Random 1 Basestation	3.844	6.211	3.532	5.467	1.683	3.845	7.186	2.051
Path 58 - Random 2 Basestation	3.879	6.844	4.322	5.791	2.653	3.879	6.597	2.799
Path 59 - Random 3 Basestation	4.025	6.539	4.29	6.171	2.549	4.025	6.243	2.585
Path 60 - Random 5 Basestation	4.671	8.3	6.703	7.742	5.138	4.672	7.926	4.782
Path 61 - Random 1 - 2 Basestation	11.615	7.006	5.13	9.235	6.152	11.616	10.761	5.187
Path 62 - Random 2 - 1 Basestation	16.497	9.058	6.078	14.752	10.373	16.498	12.927	10.402
Path 63 - Random 3 - 5 Basestation	7.198	15.908	14.033	9.622	12.781	7.198	13.99	6.954
Path 64 - Random 5 - 3 Basestation	8.927	7.08	5.498	9.695	6.224	8.927	8.42	5.526
Average	9.058	7.571	6.675	8.28	5.277	9.058	8.708	5.144
Std	4.408	3.473	3.645	3.162	3.564	4.408	3.801	3.439
Rank	7	4	3	5	2	7	6	1

Table V.16: Comparison of estimation error for paths 21 - 64

	Less than No Forgettting			ActSimple less than		
	n	v	p	n	v	p
No Forgetting	——	——	——	44	71.0	<**0.0001**
Queue Static	35	203.0	**0.0337**	44	34.0	<**0.0001**
Queue Dynamic	34	108.0	**0.0004**	44	118.0	<**0.0001**
Random	36	246.0	0.0881	44	21.0	<**0.0001**
ACT-R	44	71.0	<**0.0001**	44	600.0	0.8897
SIMPLE	38	609.5	0.9999	44	71.0	<**0.0001**
SIMPLE Update	31	221.0	0.3042	44	30.0	<**0.0001**
ActSimple	44	71.0	<**0.0001**	——	——	——

The reduction in estimation error when compared to No Forgetting (column *Less than No Forgetting*) was significant for Queue Static ($v = 203.0$, $p = 0.0337$), Queue Dynamic ($v = 108.0$, $p = 0.0004$), ACT-R ($v = 71.0$, $p < 0.0001$), and ActSimple ($v = 71.0$, $p < 0.0001$). The results for Random forgetting ($v = 246.0$, $p = 0.0881$), SIMPLE ($v = 609.5$, $p = 0.9999$), and SIMPLE Update ($v = 221.0$, $p = 0.3042$) were not significant. These results suggest than the ability for Human-Inspired Forgetting algorithms to improve performance relative to No Forgetting was not limited to only the original twenty paths tested in Chapter V.1.3. Despite changing multiple path and environmental properties, four of the forgetting algorithms were able to generate less estimation error than No Forgetting. Forgetting algorithms may possess the ability to improve system accuracy even for conditions for which they were not directly optimized.

The second Wilcoxon signed rank test (column *ActSimple less than*) tested if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms. The results were significant for No Forgetting ($v = 71.0$, $p < 0.0001$), Queue Static ($v = 34.0$, $p < 0.0001$), Queue Dynamic ($v = 118.0$, $p < 0.0001$), Random (21.0, $p < 0.0001$), SIMPLE ($v = 71.0$, $p < 0.0001$), and SIMPLE Update ($v = 30.0$, $p < 0.0001$). However, comparing ActSimple to ACT-R was not significant ($v = 600.0$, $p = 0.8897$). These results suggest that trace-based decay may play a role in the ability for forgetting algorithms to improve system accuracy.

The effect of dynamism on the results was explored by using a single-sided Wilcoxon rank sum test to determine if No Forgetting and the forgetting algorithms generated less estimation error on static paths as compared to dynamic paths. Results from this test are presented in Table V.17. The evaluation was conducted in an identical fashion to the test presented in Chapter V.1.3, and was performed with $\alpha = 0.05$, $n_A = 20$, and $n_B = 24$.

The difference in average estimation error between static and dynamic paths was significant for No Forgetting and all of the forgetting algorithms. No Forgetting ($w = 0.0$, $p < 0.0001$), Queue Static ($w = 104.0$,

Table V.17: The effects of dynamism on estimation error for paths 21 - 64

	w	p
No Forgetting	0.0	<**0.0001**
Queue Static	104.0	**0.0005**
Queue Dynamic	123.0	**0.0026**
Random	7.0	<**0.0001**
ACT-R	105.0	**0.0006**
SIMPLE	0.0	<**0.0001**
SIMPLE Update	29.0	<**0.0001**
ActSimple	113.0	**0.0011**

$p = 0.0005$), Queue Dynamic ($w = 123.0$, $p = 0.0026$), Random ($w = 7.0$, $p < 0.0001$), ACT-R ($w = 105.0$, $p = 0.0006$), SIMPLE ($w = 0.0$, $p < 0.0001$), SIMPLE Update ($w = 29.0$, $p < 0.0001$), and ActSimple ($w = 113.0$, $p = 0.0011$) each generated less estimation error when applied to only static paths as compared to being applied to only dynamic paths. These results suggest that static conditions may provide fewer challenges to mobile robots or are possibly more forgiving of poor performance.

Dynamism may affect the relative estimation error generated by No Forgetting and the forgetting algorithms. Four single-sided Wilcoxon signed rank tests were performed to test the difference in estimation error that results from applying No Forgetting and the forgetting algorithms to just static paths and to just dynamic paths. Results from these tests are presented in Table V.18. The first test (*Less than No Forgetting - Static*) evaluated if the forgetting algorithms listed in the left-most column generated less estimation error when only results from the twenty static paths were considered. The second test (*Less than No Forgetting - Dynamic*) evaluated if the forgetting algorithms generated less estimation error than No Forgetting when only results from the twenty four dynamic paths were included. The third test (*ActSimple less than - Static*) evaluated if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms when applied to only the static paths. The fourth test (*ActSimple less than - Dynamic*) evaluated if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms when applied to only dynamic paths. The α value was 0.05 for each test.

The first Wilcoxon signed rank test in Table V.18 (column *Less than No Forgetting - Static*) evaluated if the forgetting algorithms generated less estimation error than No Forgetting on the new static paths. Only the results from ActSimple ($v = 45.0$, $p = 0.0120$) and ACT-R ($v = 40.0$, $p = 0.0068$) were significant. However, the results from the second Wilcoxon signed rank test (column *Less than No Forgetting - Dynamic*) showed a different trend. The results from all forgetting algorithms except for SIMPLE were significant. Queue Static ($v = 18.0$, $p = 0.0002$), Queue Dynamic ($v = 16.0$, $p = 0.0002$), Random ($v = 13.0$, $p = 0.0001$), ACT-R ($v = 11.0$, $p < 0.0001$), SIMPLE Update ($v = 34.0$, $p = 0.0062$), and ActSimple ($v = 3.0$, $p <$

Table V.18: Estimation error performance on static and dynamic paths for paths 21 - 64

	Less than No Forgettting						ActSimple less than					
	Static			Dynamic			Static			Dynamic		
	n	v	p	n	v	p	n	v	p	n	v	p
No Forgetting	——	——	——	——	——	——	20	45.0	**0.0120**	24	3.0	<**0.0001**
Queue Static	15	120.0	1.0000	20	18.0	**0.0002**	20	10.0	<**0.0001**	24	8.0	<**0.0001**
Queue Dynamic	14	74.0	0.9137	20	16.0	**0.0002**	20	36.0	**0.0042**	24	30.0	**0.0001**
Random	16	136.0	1.0000	20	13.0	**0.0001**	20	10.0	<**0.0001**	24	3.0	<**0.0001**
ACT-R	20	40.0	**0.0068**	24	11.0	<**0.0001**	20	165.0	0.9893	24	144.0	0.4387
SIMPLE	17	136.5	0.9988	21	178.5	0.9872	20	45.0	**0.0120**	24	3.0	<**0.0001**
SIMPLE Update	12	76.0	0.9995	19	34.0	**0.0062**	20	15.0	**0.0001**	24	3.0	<**0.0001**
ActSimple	20	45.0	**0.0120**	24	3.0	<**0.0001**	——	——	——	——	——	——

0.0001) each generated less estimation error than No Forgetting when processing only the twenty four new dynamic paths. These results suggest than Human-Inspired Forgetting based forgetting algorithms may be an effective approach to improving system accuracy in dynamic domains, but trace-based decay may provide some forgetting algorithms the ability to improve performance in certain static environments as well.

The third Wilcoxon signed rank test in Table V.18 (column *ActSimple less than - Static*) evaluated if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms when only applied to the new static paths. Results from No Forgetting and all of the other forgetting algorithms, except ACT-R ($v = 165.0$, $p = 0.9893$), were significant. No Forgetting ($v = 45.0$, $p = 0.0120$), Queue Static ($v = 10.0$, $p < 0.0001$), Queue Dynamic ($v = 36.0$, $p = 0.0042$), Random ($v = 10.0$, $p < 0.0001$), SIMPLE ($v = 45.0$, $p = 0.0120$), and SIMPLE Update ($v = 15.0$, $p = 0.0001$) each generated more estimation error than ActSimple when applied to only the new static paths. The results from the fourth Wilcoxon signed rank test (column *ActSimple less than - Dynamic*) showed a similar trend. Except for ACT-R ($v = 144.0$, $p = 0.4387$), the comparisons against No Forgetting and the remaining forgetting algorithms were significant. No Forgetting ($v = 3.0$, $p < 0.0001$), Queue Static ($v = 8.0$, $p < 0.0001$), Queue Dynamic ($v = 30.0$, $p = 0.0001$), Random ($v = 3.0$, $p < 0.0001$), SIMPLE ($v = 3.0$, $p < 0.0001$), and SIMPLE Update ($v = 3.0$, $p < 0.0001$) were all found to generate more estimation error when processing the new dynamic paths. These results show ActSimple improving system accuracy in both static and dynamic conditions as compared to No Forgetting and all of the forgetting algorithms except ACT-R. ActSimple and ACT-R both incorporate trace-based decay and these results suggest that this method of implementing forgetting in robotic systems may be an effective approach to improving performance across a wide range of realistic conditions that could be presented to robotic systems.

V.3.3 Summary

The forgetting algorithms and No Forgetting were tested with a set of 44 new paths, labeled paths 21 - 64. These paths possessed different trajectories, basestation configurations, and number of basestation configuration changes, but the forgetting algorithms (except for SIMPLE) were able to generate less average absolute estimation error than No Forgetting.

While the relative estimation performance of the forgetting algorithms varied from the original twenty paths, the ActSimple algorithm produced the smallest estimation error for both sets of paths (Tables V.6 and V.15). SIMPLE Update, which generated the second best performance for paths 1 - 20, only realized the sixth best estimation error for the new paths. These results suggest that forgetting algorithms embodying Human-Inspired Forgetting possess the ability to improve accuracy despite changing environmental conditions and task properties. However, the relative benefits of the forgetting algorithms may not be constant. The ActSimple algorithm appears to capitalize on its combination of trace-based decay and similarity-interference to generate improved filtering capabilities. The SIMPLE Update algorithm generated less estimation error than ACT-R when applied to paths 1 - 20 and ACT-R averaged less estimation error than SIMPLE Update when applied to paths 21 - 64, but ActSimple realized the least amount of estimation error for both groups of paths.

While the ActSimple algorithm generated the smallest amount of average absolute estimation error, it possess the largest number of parameters. The additional parameters in ActSimple may lengthen the amount of time required to optimize the algorithm for a new task or domain, but the results suggest that the extra parameters may not cause over-learning by the algorithm. A comparison of each algorithm's complexity (number of parameters) and the algorithm's resultant average absolute estimation error is presented in Figure V.6. This graph uses the same key as Figure V.2, but presents the combined results of all 44 paths (paths 21 - 64), the twenty new static paths, and the 24 new dynamic paths.

No Forgetting and the forgetting algorithms each generated less estimation error when operating with just the static paths as compared to the full set of paths. This trend is identical to the trend generated by paths 1 - 20. No Forgetting and the forgetting algorithms also generated more error when applied to the dynamic paths, than when applied to the full set of 44 paths. Dynamism appears to result in more challenging conditions for both No Forgetting and the forgetting algorithms, although the magnitude of the effects do not appear to be consistent. The estimation error difference from applying a forgetting algorithm (or No Forgetting) to only static paths versus only dynamic paths varies across No Forgetting and the forgetting algorithms. No Forgetting and SIMPLE resulted in the largest performance differences.

ACT-R and ActSimple generated less average absolute estimation error than No Forgetting, when only applied to the static paths. This result suggests that at least under certain conditions, Human-Inspired Forgetting

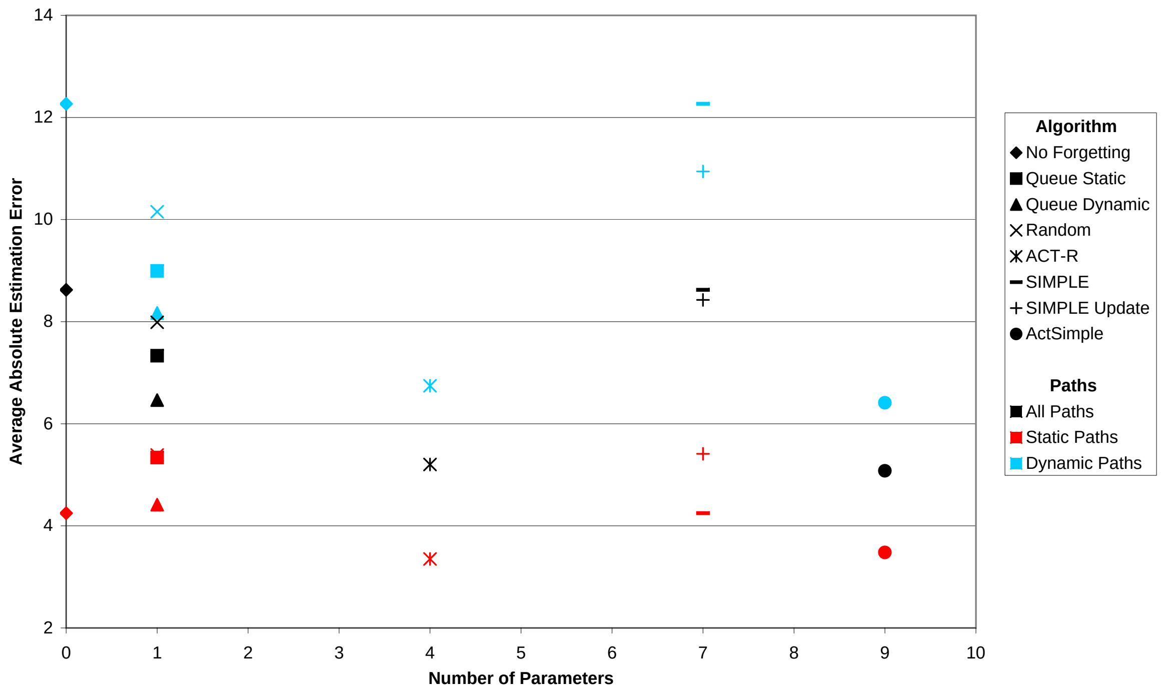


Figure V.6: Number of parameters vs. average absolute estimation error for paths 21 - 64

has the potential to improve performance in static environments. The filtering afforded by these algorithms may allow robotic systems to better reject noise inherent in complex domains, resulting in greater accuracy.

The average number of recallable readings metric when No Forgetting and the forgetting algorithms were applied to paths 21- 64 was also collected (Table V.14). The rank ordering of the number of recallable readings resulting from each forgetting algorithm and No Forgetting was identical to when the forgetting algorithms were applied to paths 1 - 20. No Forgetting averaged the largest number of recallable readings, Random forgetting averaged the least, and ActSimple averaged the second smallest number of recallable readings. The filtering provided by a forgetting algorithm is only beneficial if the resultant estimation error either improves or only increases slightly. A graph comparing the average number of recallable readings and the average absolute estimation error from No Forgetting and the forgetting algorithms is presented in Figure V.7.

The average number of recallable readings generated by No Forgetting and each forgetting algorithm was smallest when only static paths were considered. As a result, the dynamic paths resulted in larger numbers of recallable readings. The difference in the number of recallable readings between static and dynamic conditions varied across the forgetting algorithms. No Forgetting and SIMPLE experienced a relatively large difference, while ActSimple and SIMPLE Update generated more consistent values (Figure V.7). When ap-

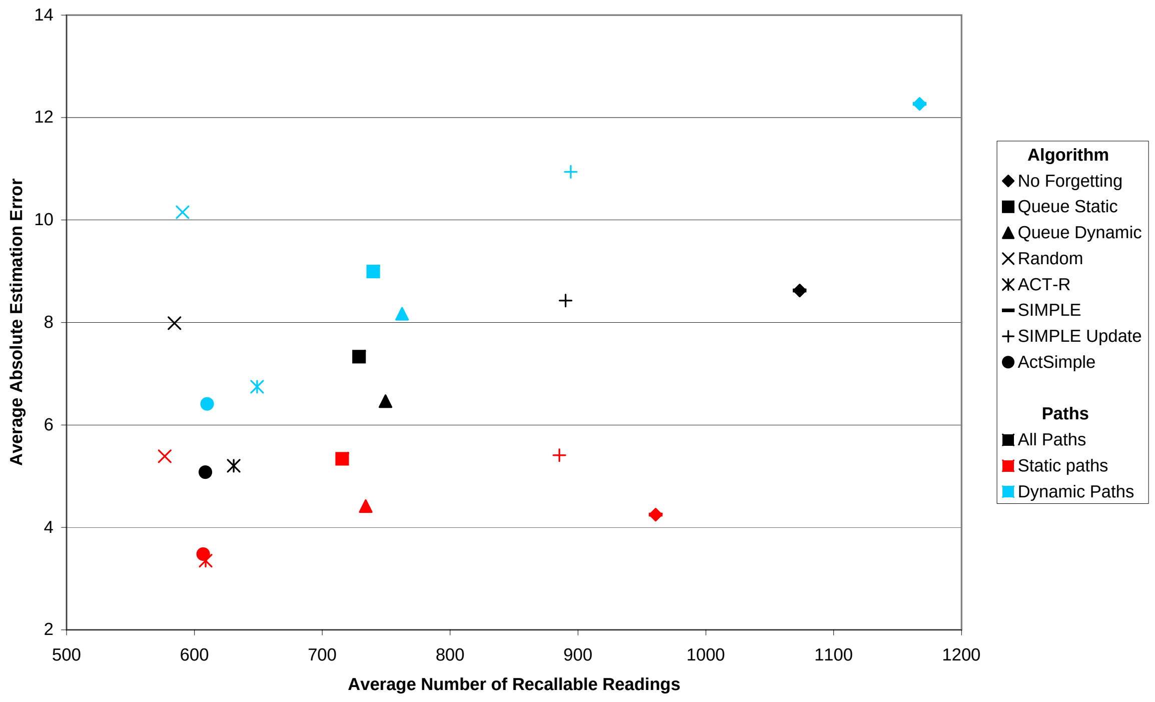


Figure V.7: Average number of recallable readings vs. average estimation error for paths 21 - 64

plied to each of the three sets of paths, ActSimple and ACT-R generated the least amount of average absolute estimation error and fewer recallable readings than No Forgetting and all other forgetting algorithms, except for Random forgetting. These results suggest that the application of Human-Inspired Forgetting algorithms may have the potential to increase accuracy, while also reducing the number of data points that require processing by potentially computationally expensive existing robotic algorithms.

V.4 Summary

This chapter presented a set of experiments where the seven forgetting algorithms described in Chapter IV were optimized and tested under three sets of conditions. Two metrics were collected, the average absolute estimation error resulting from a robot estimating the WiFi signal strength at each point within a simulated environment, and the average number of recallable WiFi signal strength readings that were presented to the existing robotic algorithms.

Results from this chapter have shown that simply providing additional data to an existing robotic algorithm may not be an effective approach to improving accuracy. After the forgetting algorithms were optimized, they were tested with a set of twenty paths. While No Forgetting generated the largest number of recallable readings, No Forgetting tied for the largest amount of average estimation error. Conversely, ActSimple filtered the second most number of readings, but still resulted in the smallest amount of average estimation error. The selectivity of an information filter (forgetting algorithm) appears critical to the effectiveness and accuracy of a system. Intelligent removal of detrimental information has the potential to improve system accuracy and reduce the volume of data requiring potentially computationally expensive processing.

Mobile robots operating in complex and dynamic domains are frequently presented with new challenges and must adjust. The tasks performed by these systems are not static and constantly adapt to changing conditions. The ability for the tested forgetting algorithms to reduce average estimation error despite small changes to task was evaluated by combing the algorithms with three additional interpolation methods. No Forgetting generally resulted in or tied for the largest amount of estimation error. Conversely, ActSimple consistently produced the smallest amount of estimation error. ActSimple and Human-Inspired Forgetting appear to degrade gracefully in the presence of small changes to task.

In addition to adapting to changing tasks, mobile robots must also perform well in the face of changing environmental conditions. The forgetting algorithms were tested with a set of 44 additional paths, comprised of new trajectories, basestation configurations, and basestation configuration changes. The relative performance of the forgetting algorithms changed when applied to these new paths, but No Forgetting once again tied for the largest amount of estimation error, while ActSimple resulted in the smallest amount of estimation error. ActSimple did not produce the smallest amount of estimation error for every single path, but on average, outperformed No Forgetting and the other forgetting algorithms. No forgetting algorithm may be capable of guaranteeing the best performance on any one specific set of condition. Average performance is often more important to mobile robots, especially if extreme performance degradations can be avoided.

The results presented in this chapter suggest that Human-Inspired Forgetting may be an effective and viable means of simultaneously improving system accuracy, while minimizing the number of points that must be processed by potentially computationally expensive existing robotic algorithms. No Forgetting resulted in at least a tie for the largest amount of estimation error, despite never filtering a signal strength reading, in nearly every tested condition. The forgetting algorithms employing Human-Inspired Forgetting, excluding SIMPLE, were able to simultaneously filter data, while improving accuracy. ActSimple was able to consistently generate the smallest amount of estimation error, while filtering the largest number of readings. These results suggest that ActSimple is an effective forgetting algorithm that may also be resistant to changes in task and environment. When operating within complex and dynamic domains, algorithms employed by mobile robots must be capable of operating across a diverse array of conditions and degrade gracefully in the presence of unexpected situations.

CHAPTER VI

Real World Testing

The real world poses many challenges that are difficult to model or account for in simulation. Robotic algorithms require real world testing to verify that unintended properties of a simulator do not mask true performance levels or unanticipated real world challenges do not negate an algorithm's benefits. This chapter presents an experiment where ActSimple and the other six forgetting algorithms were applied to real world WiFi readings. The experiment was conducted in an identical manner to previous simulation evaluations (Chapter V), except that a real robot collected the WiFi signal data from an outdoor environment and a collection of new paths and basestation configurations were employed.

VI.1 Experimental Design

Before the forgetting algorithms could be applied to any real world WiFi data, truth information was required in order to calculate the estimation error values. The real world is presently too difficult to model completely, forcing truth information to be collected empirically. Initially, the experiment was to be conducted on Vanderbilt University's campus, but the lack of control and repeatability over WiFi quality in a city environment made this approach impractical. WiFi interference can result from numerous sources, including cell phones, microwaves, computers, air conditioners, other WiFi equipment, and heavy machinery. While the forgetting algorithms were tasked with removing the effects of this interference during testing, uncontrolled interference prevents the generation of truth information guaranteed to be consistent and valid throughout the experiment.

Instead, the experiment was performed at the Colson Hollow group camp grounds, an area located within the Land Between the Lakes national recreation area. Colson Hollow consists of a large grassy field surrounded by forests on three sides and a river on the fourth. The campgrounds are isolated from many sources of WiFi interference, while providing a large open experimental area. Power lines are present at Colson Hollow and a radio transmitter is within a few miles of the location, but the reliability of collected truth data was greater than if readings were collected on Vanderbilt's campus.

Similar to the simulation-based optimization experiment (Chapter V), the real world experiment involved two WiFi basestation configurations and a large number of WiFi sampling locations, see Figure VI.1. Due to logistical constraints, changes to the environment were made. The sampling area was 300 ft. by 300 ft. and samples were taken along a ten by ten grid. The WiFi basestations were situated 25 ft. outside of the sampling region and were placed on approximately 5 ft. tall wooden stands.

A Pioneer 3DX robot equipped with a WiFi enabled laptop computer recorded the readings. While the

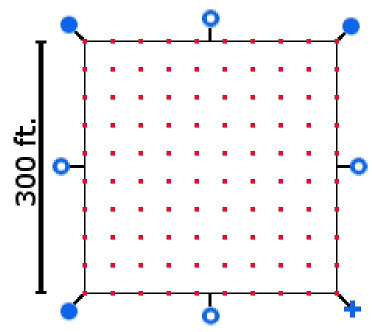


Figure VI.1: Real world experiment map. The red points represent sample locations. The blue circles represent WiFi basestations that were always present in both configurations, blue rings represent WiFi basestations present in the seven basestation configuration, and the blue cross represents a basestation present only in the four basestation configuration.

robot was running during data collection, the ground conditions prevented the robot from driving through the environment. Sampling was performed by manually placing the robot at each location. Samples were collected with a Dell TrueMobile 1150 Series Mini PCI card (Dell, 2003) and the NetStumbler 0.4.0 (NetStumbler, 2002) WiFi monitoring program. All samples required for a particular sampling location and basestation configuration were collected at one time for a 6 minute interval with a frequency of up to 2 Hz. Actual sampling frequency was dependent on individual WiFi basestation response times.

During the evaluation, readings were selected randomly from the samples recorded for a particular location and basestation configuration. This approach was employed to minimize the effects of any interference present in the environment and to increase the efficiency of data collection. The Colson Hollow location minimized the effects of persistent, low frequency noise. In the ideal case, no noise would be present and the process of collecting WiFi readings in batches would not affect the results. However, if noise was present, randomly selecting readings would minimize these effects. The collection of readings in batches permitted a larger number of paths to be tested with fewer recorded samples.

The NetStumbler application created sampling log files for each combination of location and basestation configuration in Wi-Scan file format. NetStumbler actively sends WiFi packets to detected basestations in order to determine the available signal level, noise level, and Signal to Noise Ratio (SNR), timestamped to the second. In the generated files, samples are recorded when available, instead of at a set frequency.

Before the data could be evaluated, the raw logs required processing, which consisted of several steps. For each second within the logging time window for a particular location and basestation combination, the reading with the highest SNR value was selected. If no readings were available for a particular second, a value of 0 dB was assigned. The NetStumbler program occasionally produced samples where both the reported signal strength and the noise level were -32618. These values are erroneous and were discarded before processing began. The SNR values were then multiplied by 2.0, incremented by 1.0, and then clipped to the range 1 to 100. This conversion process was performed so that the results are on the same scale as the results generated during optimization. While the clipping process does result in some lost information, the robot's primary goal is to locate problematic areas possessing poor WiFi capabilities. As a rule of thumb, a SNR over 40 dB is considered to be very good. The clipping process only affects readings with a SNR over 50 dB, readings already indicative of areas not experiencing degraded WiFi communications. As the final processing step, a Max 2 filter was applied to the data. Resulting from the active WiFi signal strength measuring technique employed by the NetStumbler program, the generated readings possessed an unnecessarily large number of minimum value readings. These extra readings resulted not from weak signal strength, but from basestations occasionally not reporting strength values at a high enough frequency to ensure valid data at every time stamp. The Max 2 filter tested each reading and if the previous reading was larger, then the current reading was changed to the value of the previous reading. The Max 2 filter only operated on the pre-filtered data, otherwise, the readings would have been monotonically increasing.

VI.2 Real World Results

All of the real world data was collected during a four day period in June, 2010. The entirety of the data was used to calculate the truth data. After the readings were processed as described in Section VI.1, the readings for each combination of location and basestation configuration were averaged. Results from this averaging process are shown in Figures VI.2a and VI.2b. In these figures, black represents areas possessing a greater SNR. The presence of the basestations in each condition can be clearly observed, although exact basestation positions are not always easy to determine. Interference appears to have slightly affected the top and right portions of the seven basestation truth map, and to a lesser extent the truth map for the four basestation configuration. This interference resulted in weaker SNR values and a greater difficulty in determining basestation positions. Definitive differences between areas of high SNR and areas with low SNR can be observed.

The forgetting algorithms were tested with a set of eighteen new paths, as the paths used during optimization did not fit in the smaller real world environment. These new paths ranged in length from 337 to 3177 steps ($\mu = 1831.278$ and $\sigma = 1032.329$) with six generated randomly. A basestation configuration change occurred in eight of the paths, at roughly the midpoint of the path's trajectory. While the real world environment is only 10 by 10, the forgetting algorithms were optimized in the larger simulation environment. When relatively short paths are processed by the forgetting algorithms, they consistently recalled nearly every reading. Six additional paths, with a length of 208 steps or less, were also tested, but since the seven forgetting algorithms and No Forgetting generated effectively identical recall and estimation error values, these

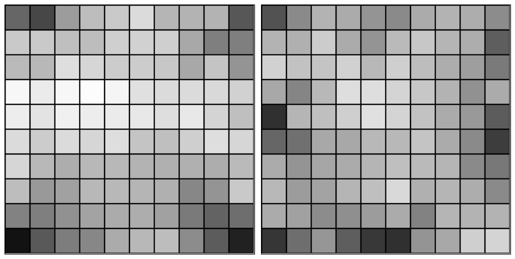

(a) 4 basestations (b) 7 basestations

Figure VI.2: Maps of real world truth values

paths were not included in the presented results. The removal of these paths did not affect the error rankings. Path instances were created by randomly selecting readings from the log files, and readings were allowed to be selected multiple times. During testing, parameterizations determined in the optimization experiment (Chapter V) were used.

Table VI.1: Real world results

		No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	Simple	Simple Update	ActSimple
Recall	Average	848.316	675.979	692.467	544.366	555.095	848.297	632.063	338.887
	Std	331.196	179.450	189.407	108.914	131.159	331.189	206.261	36.805
	Rank	1st	4th	3rd	7th	6th	2nd	5th	8th
Error	Average	4.588	3.084	3.289	3.855	3.840	4.588	2.382	3.069
	Std	3.182	1.940	2.277	2.122	2.151	3.1818	1.038	0.950
	Rank	7th	3rd	4th	6th	5th	7th	1st	2nd

The overall recallable reading counts and estimation error values are presented in Table VI.1. Ranking values were computed after rounding was performed to minimize the effects of insignificant noise. No Forgetting generated the largest average number of recallable readings, with SIMPLE averaging only 0.019 fewer. Both No Forgetting and the SIMPLE algorithm maintained similar recallable reading count standard deviations (331.196 and 331.189, respectively), which were 124.928 higher than the next highest standard deviation (SIMPLE Update's standard deviation of 206.261). ActSimple averaged only 338.887 recallable readings, 509.429 less than No Forgetting, while maintaining a standard deviation of only 36.805. All other forgetting algorithms averaged a greater number of recallable readings than ActSimple.

The performance differences between each forgetting algorithm and No Forgetting are presented in Table VI.2. The Outperformed row represents the number of paths where a forgetting algorithm outperformed No Forgetting, while the Underperformed row displays the number of paths where the forgetting algorithm generated greater estimate error. The Outperformed Ave. and Underperformed Ave. rows provide the average error deviation for paths where the forgetting algorithm outperformed or underperformed No Forgetting,

Table VI.2: Forgetting performance deviations from No Forgetting

Metric	Queue Static	Queue Dynamic	Random	ACT-R	Simple	Simple Update	ActSimple
Average Deviation	1.504	1.299	0.733	0.747	0	2.205	1.518
Deviation Std.	2.386	2.176	1.53	1.104	0	2.639	2.542
Outperformed	8	8	8	9	8	9	8
Outperformed Ave.	3.594	2.985	2.088	1.692	0	4.454	4.209
Underperformed	4	3	6	7	7	5	10
Underperformed Ave.	0.42	0.169	0.586	0.254	0	0.079	0.634
Equal Performance	6	7	4	2	3	4	0

respectively.

Despite averaging far fewer recallable readings, ActSimple's average estimation error (3.069) was 1.518 less than No Forgetting. Although ActSimple also maintained the only estimation standard deviation under 1.0 (0.950), SIMPLE Update averaged lower estimation error (2.382). Queue Static maintained the third best average estimation error (3.084).

All forgetting algorithms outperformed No Forgetting on eight of the eighteen paths except for ACT-R and SIMPLE Update, which both generated smaller error on nine paths. The number of paths where the forgetting algorithms underperformed No Forgetting was more variable, ranging from three to ten paths. ActSimple underperformed No Forgetting ten times, while ACT-R underperformed seven times and SIMPLE Update underperformed five times. The outperformed and underperformed values for SIMPLE are not reliable, since the largest estimation error performance difference between SIMPLE and No Forgetting was less than 0.001.

Trends were also observed in the individual path data. Basestation configuration changes generated substantial performance variations in the forgetting algorithms and No Forgetting. When basestation configuration changes were not present, the worst estimation error generated by No Forgetting was 3.614, but the best error produced for a path with a basestation configuration change was 7.547. Excluding SIMPLE, every forgetting algorithm outperformed No Forgetting when a basestation configuration change occurred. Again ignoring SIMPLE, SIMPLE Update and ACT-R were the only forgetting algorithms to outperform No Forgetting on static paths. However, both forgetting algorithms only outperformed No Forgetting on one path each and by a margin less than 0.001.

The performance properties exhibited by No Forgetting and the forgetting algorithms when no basestation configuration change occurred can partially be explained by the method used to generate the truth data. Readings for each pairing of location and basestation configuration were averaged together to form the truth value for each combination. During paths when no basestation configuration change occurred, No Forgetting benefited from being able to recall every reading to help average out noise present in the readings, essentially mimicking the truth data generation procedure. If truth values were available by a means other than a weighted average, the performance differences along static versus dynamic paths may decrease.

The statistical significance of the relative estimation error performance results from Table VI.1 was tested with two Wilcoxon signed rank tests and are presented in Table VI.3. These tests were performed in a similar fashion to those presented in Chapter V.1.3. The first test, labeled *Less than No Forgetting*, evaluated if the forgetting algorithms listed in the left-most column generated less average estimation error than No Forgetting. The second test, labeled *ActSimple less than*, evaluated if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms. Both tests were single-sided and α was 0.05. Significant results are indicated in bold and the individual pairings that were not performed are marked with a —— symbol.

Table VI.3: Comparison of estimation error for real-world paths

	Less than No Forgetting			ActSimple less than		
	n	v	p	n	v	p
No Forgetting	——	——	——	18	55.0	0.0982
Queue Static	12	10.0	**0.0105**	18	102.0	0.7659
Queue Dynamic	11	6.0	**0.0068**	18	101.0	0.7525
Random	14	26.0	0.0520	18	50.0	0.0649
ACT-R	16	35.0	**0.0467**	18	55.0	0.0982
SIMPLE	15	57.0	0.4452	18	55.0	0.0982
SIMPLE Update	14	20.0	**0.0209**	18	171.0	1.0000
ActSimple	18	55.0	0.0982	——	——	——

The reduction in estimation error when compared to No Forgetting (column *Less than No Forgetting*) was significant for Queue Static ($v = 10.0$, $p = 0.0105$), Queue Dynamic ($v = 6.0$, $p = 0.0068$), ACT-R ($v = 35.0$, $p = 0.0467$), and SIMPLE Update ($v = 20.0$, $p = 0.0209$). Results from Random, SIMPLE, and ActSimple were not significant. The Wilcoxon signed rank test labeled *ActSimple less than* evaluated if ActSimple generated less estimation error. None of the results were significant, despite ActSimple generating the second smallest average estimation error (Table VI.1).

SIMPLE Update averaged the smallest amount of estimation error when applied to the real-world paths (Table VI.2). A single-sided Wilcoxon signed rank test was conducted to determine if SIMPLE Update generated less estimation error than ActSimple on these paths and the results were significant ($n = 18$, $v = 0$, $p < 0.0001$). SIMPLE Update outperformed ActSimple when applied to this particular set of testing conditions.

The effects of dynamism on the results was explored by using a single-sided Wilcoxon rank sum test to determine if No Forgetting and the forgetting algorithms generated less estimation error on static paths as compared to dynamic paths. This test was conducted in an identical fashion to the tests presented in

Chapter V, and the results are presented in Table VI.4. The α was set to 0.05, while $n_A = 10$, and $n_B = 8$.

Table VI.4: The effects of dynamism on estimation error for real-world paths

	w	p
No Forgetting	0.0	<**0.0001**
Queue Static	11.0	**0.0043**
Queue Dynamic	8.0	**0.0015**
Random	0.0	<**0.0001**
ACT-R	0.0	<**0.0001**
SIMPLE	0.0	<**0.0001**
SIMPLE Update	17.0	**0.0217**
ActSimple	8.0	**0.0015**

The difference in average estimation error between static and dynamic paths was significant for No Forgetting and all of the forgetting algorithms. No Forgetting and the forgetting algorithms, on average, found the real-world paths containing a basestation configuration change to be more challenging.

The presence of basestation configuration changes may alter the relative estimation error performance of No Forgetting and the forgetting algorithms. Four single-sided Wilcoxon signed rank tests were performed to evaluate the difference in estimation error that results from applying No Forgetting and the forgetting algorithms to static paths and dynamic paths. Results from these tests are presented in Table VI.5. The first test (*Less than No Forgetting - Static*) evaluated if the forgetting algorithms listed in the left-most column generated less estimation error than No Forgetting when only results from the static paths were considered. The second test (*Less than No Forgetting - Dynamic*) evaluated if the forgetting algorithms generated less estimation error than No Forgetting on the dynamic paths. The third test (*ActSimple less than - Static*) evaluated if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms when applied to only the static paths. The fourth test (*ActSimple less than - Dynamic*) evaluated if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms when applied to the dynamic paths. The α was 0.05 for each test.

The Wilcoxon signed rank test in Table VI.5 for the *Less than No Forgetting - Static* condition evaluated if the forgetting algorithms generated less estimation error than No Forgetting on static paths. None of the results were significant. The results of the Wilcoxon signed rank test for the *Less than No Forgetting - Dynamic* condition showed a different trend. The results from all of the forgetting algorithms, except SIMPLE, were found to be significant ($v = 0.0$, $p = 0.0039$). These results suggest that Human-Inspired Forgetting based forgetting algorithms may be an effective approach to improving system accuracy in dynamic real-world conditions.

The Wilcoxon signed rank test for the *ActSimple less than - Static* condition evaluated if ActSimple

Table VI.5: Estimation error performance on real-world static and dynamic paths

	Less than No Forgetting						ActSimple less than					
	Static			Dynamic			Static			Dynamic		
	n	v	p	n	v	p	n	v	p	n	v	p
No Forgetting	——	——	——	——	——	——	10	55.0	1.0000	8	0.0	**0.0039**
Queue Static	4	10.0	1.0000	8	0.0	**0.0039**	10	52.0	0.9971	8	10.0	0.1562
Queue Dynamic	3	6.0	1.0000	8	0.0	**0.0039**	10	55.0	1.0000	8	8.0	0.0977
Random	6	21.0	1.0000	8	0.0	**0.0039**	10	50.0	0.9932	8	0.0	**0.0039**
ACT-R	8	35.0	0.9961	8	0.0	**0.0039**	10	55.0	1.0000	8	0.0	**0.0039**
SIMPLE	8	25.0	0.8438	7	7.0	0.1484	10	55.0	1.0000	8	0.0	**0.0039**
SIMPLE Update	6	20.0	0.9844	8	0.0	**0.0039**	10	55.0	1.0000	8	36.0	1.0000
ActSimple	10	55.0	1.0000	8	0.0	**0.0039**	——	——	——	——	——	——

generated less estimation error than No Forgetting and the other forgetting algorithms when only applied to the static paths. None of the results were significant. The Wilcoxon signed rank test for the *ActSimple less than - Dynamic* condition tested if ActSimple generated less estimation error on dynamic paths. Results were significant for No Forgetting, Random, ACT-R, and SIMPLE ($v = 0.0$, $p = 0.0039$), and not significant for Queue Static, Queue Dynamic, and SIMPLE Update.

The real-world paths and environment differed greatly from the paths and environment used in Chapter V and the overall average estimation error across No Forgetting and the forgetting algorithms dropped from 7.314 to 3.5869. Additionally, ActSimple averaged the second smallest amount of estimation error and maintained the smallest estimation error standard deviation (0.950). These results suggest that ActSimple may have been able to minimize the occurrence and severity of very poor performance, but the reduction in average estimation error for No Forgetting and the forgetting algorithms may have limited the ability for ActSimple to outperform No Forgetting and the other forgetting algorithms.

VI.3 Discussion

The effectiveness of Human-Inspired Forgetting was explored by applying No Forgetting and the forgetting algorithms to real-world data. A robot was used to collect WiFi signal strength readings from a real-world environment, and these readings were used to create path instances to test the forgetting algorithms. Results from this experiment suggest Human-Inspired Forgetting has the potential to improve system accuracy in real-world environments, and especially within dynamic real-world environments.

Each forgetting algorithm, except SIMPLE, averaged less estimation error than No Forgetting when applied to the real-world data, while also reducing the average number of recallable readings. The tested environment was very different from the conditions used to originally optimize the algorithms, but the forgetting

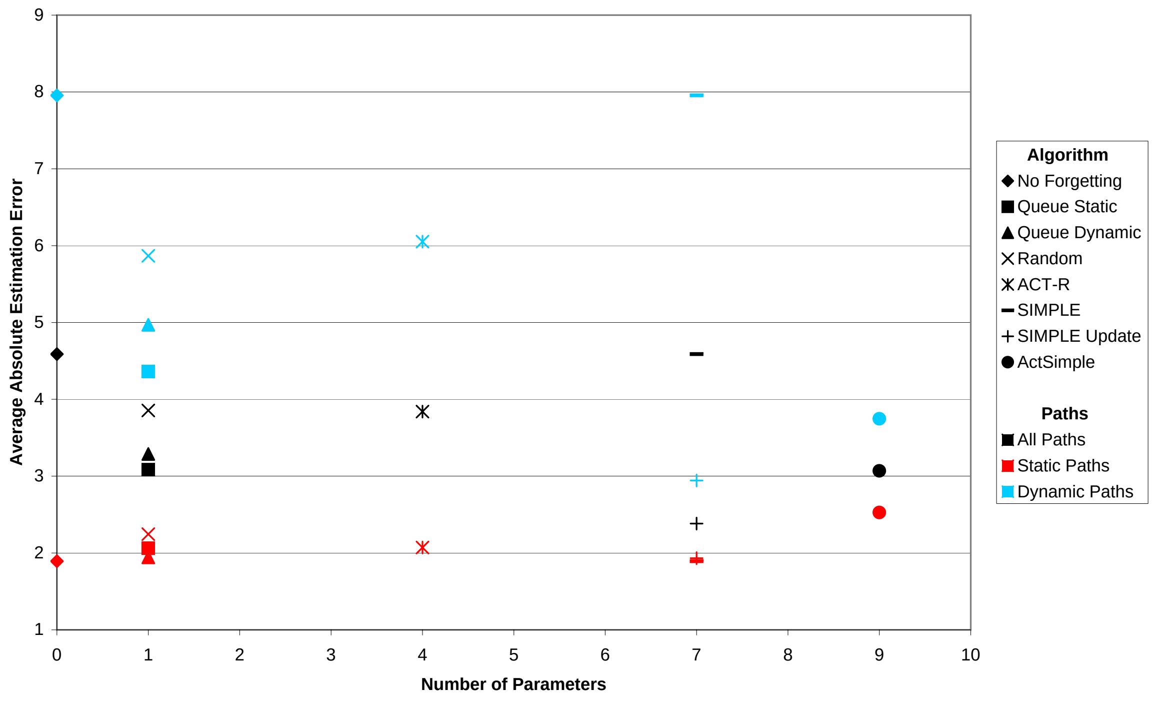


Figure VI.3: Number of parameters vs. average absolute estimation error for real world paths

algorithms still managed to increase system accuracy. While the SIMPLE Update algorithm averaged less estimation error, ActSimple generated the second smallest average estimation error and resulted in the fewest number of recallable readings.

The effects of algorithm complexity are shown in Figure VI.3, which plots an algorithm's number of parameters versus its resultant average estimation error. No Forgetting and each forgetting algorithm is represented by a unique symbol and color indicates which paths were included in the results. The SIMPLE Update and ActSimple algorithms are two of the most complex algorithms, based on parameter count, but they averaged the least amount of estimation error on dynamic paths and the full set of eighteen paths. These two algorithms also resulted in the smallest differences between performance on static paths and dynamic paths. No Forgetting and SIMPLE resulted in the least amount of estimation error on static paths but the largest amount of estimation error on dynamic paths. However, the average estimation error difference between real-world static and dynamic paths for No Forgetting and SIMPLE was smaller than the differences generated by No Forgetting and SIMPLE on the simulated paths (Figures V.2 and V.6). Many robotic systems require consistent performance. Human-Inspired Forgetting based forgetting algorithms may not improve accuracy for every single set of conditions, but the results suggest Human-Inspired Forgetting may allow robots to minimize severe accuracy degradations while improving average accuracy.

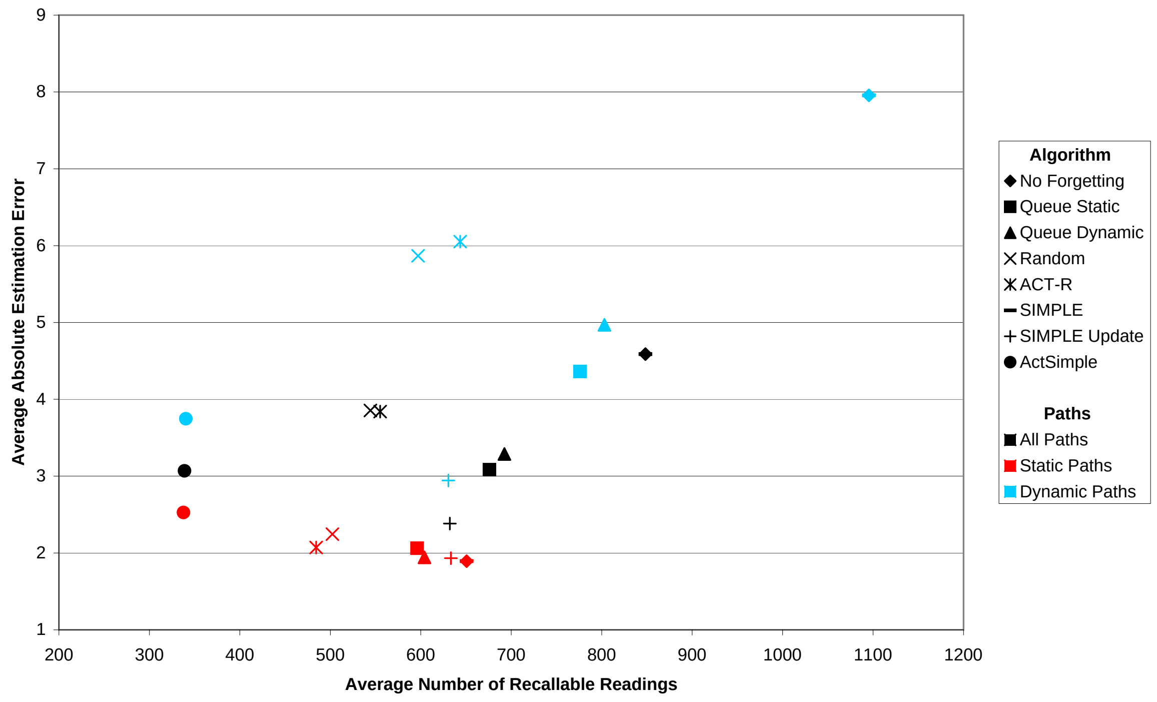


Figure VI.4: Average number of recallable readings vs. average estimation error for real world paths

A comparison between the average amount of estimation error and the average number of recallable readings that resulted from No Forgetting and each forgetting algorithm is presented in Figure VI.4. This graph uses the same key as Figure VI.3. Unlike the results generated with the simulated path data (Figures V.3 and V.7), ActSimple averaged the least number of recallable readings for the full set of paths, static paths, and dynamic paths. No Forgetting and SIMPLE resulted in the largest number of recallable readings for all three sets of paths. SIMPLE Update averaged less estimation error than ActSimple, but resulted in almost twice as many recallable readings. Mobile robots are frequently tasked with operating under tight timing constraints. These results suggest the ActSimple algorithm may be capable of improving system accuracy, while simultaneously reducing the volume of data requiring processing by computationally expensive existing robotic algorithms.

The relative benefits of forgetting depend on the forgetting algorithm, its parameterization, and compatibility with the task and environment. Each forgetting algorithm was optimized under simulated conditions differing greatly from the real-world environment employed in this chapter. However, the forgetting algorithms were still able to simultaneously reduce estimation error and the number of recallable readings. These results suggest Human-Inspired Forgetting based forgetting algorithms may be relatively resilient to changing environmental conditions. As many realistic challenges facing mobile robots constantly evolve or are difficult to model, the ability to minimize the need to frequently re-optimize may be paramount in applying forgetting to real world situations.

CHAPTER VII

Performance with Other Noise Distributions

During the optimization experiments (Chapter V), only a single noise distribution was tested. During the real world experiment (Chapter VI), an alternate noise distribution was present in the environment. Differences in the noise distributions may have lead to the performance deviations observed between the real world results and those acquired from the optimization experiment (Chapter V). This chapter describes an experiment that was conducted to explore the effects of applying the forgetting algorithms in the presence of new noise distributions. During this experiment, the seven forgetting algorithms and No Forgetting were tested with the simulated robot traveling the 64 paths described in Chapters IV and V. For each new noise distribution, the estimation error was calculated in an identical fashion as in previous experiments.

VII.1 New Noise Distributions

In addition to the original noise distribution, 21 new discrete noise distributions were tested, comprising six separate distribution families. Noise distribution families consisted of distributions based on Gaussian, rounded Gaussian, impulse, Laplace, and sinc distributions as well as No Noise. All distributions are centered at 0 and range from -100 to 100.

Original Noise Distribution: The original noise distribution's Probability Density Function (PDF) is defined by Equation IV.10 (repeated below). This distribution has only nine values with a probability greater than 0 and maintains a large probability at its center, as can be seen in Figure VII.1

$$p(x) = \begin{cases} 0.62 & x = 0 \\ 0.1 & |x| = 10 \\ 0.05 & |x| = 20 \\ 0.03 & |x| = 40 \\ 0.01 & |x| = 60 \\ 0 & \text{else} \end{cases} \tag{IV.10}$$

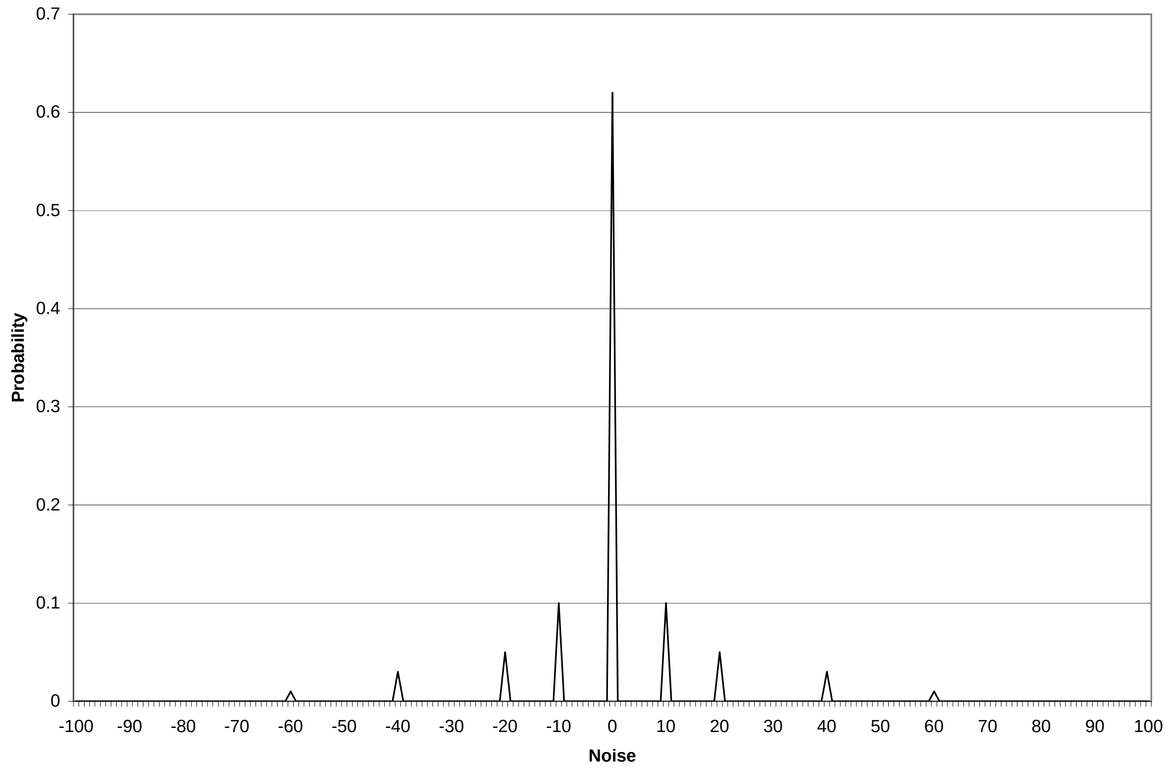


Figure VII.1: Original noise distribution PDF

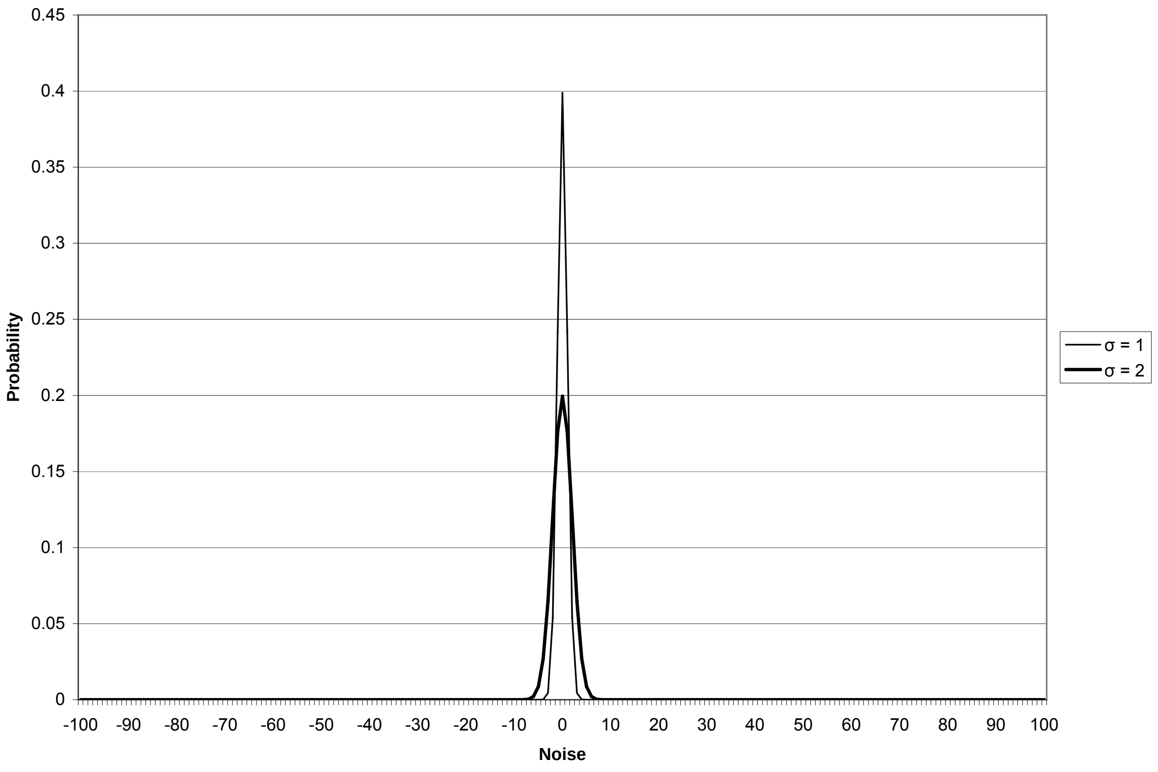


Figure VII.2: Discrete Gaussian PDF

Discrete Gaussian: The discrete Gaussian distribution PDFs are characterized by Equation VII.1 and the distributions with $\sigma = 1$ and $\sigma = 2$ are shown in Figure VII.2. Six discrete Gaussian noise distributions were tested in this experiment with σ values of 1, 1.25, 1.5, 2, 5, and 15.

$$\begin{aligned} \text{Gaussian}(x,\sigma) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \\ p(x,\sigma) &= \frac{\text{Gaussian}(x,\sigma)}{\sum\limits_{i=-100}^{100} \text{Gaussian}(\text{i},\sigma)} \end{aligned} \tag{VII.1}$$

Rounded Discrete Gaussian: The rounded discrete Gaussian family of noise distributions were inspired by the discrete Gaussian and are calculated using Equation VII.2. Unlike the discrete Gaussian, the rounded discrete Gaussian has only non-zero values that are multiples of an integer parameter, *r*. The tested rounded discrete Gaussian noise distributions have the following σ, r parameter sets: $\{5, 3\}$, $\{5, 5\}$, $\{5, 11\}$, $\{15, 3\}$, $\{15, 5\}$, and $\{15, 11\}$. Figure VII.3 contrasts the two rounded discrete Gaussian distributions $\{5, 3\}$ and $\{5, 11\}$ with the discrete Gaussian of $\sigma = 5$. The two rounded Gaussian distributions have fewer values possessing a non-zero probability than the discrete Gaussian. However, each of the non-zero probability values in the rounded Gaussian distributions has a greater probability than the equivalent value in the discrete Gaussian distribution. Figure VII.4 contrasts the two rounded discrete Gaussian distributions $\{15, 3\}$ and $\{15, 11\}$ with the discrete Gaussian with $\sigma = 15$. The trends when $\sigma = 15$ are similar to when $\sigma = 5$.

$$\begin{aligned}
\text{round}(x,r) &= r\,\lfloor x/r\rceil \\
\text{rounded\_Gaussian}(x,\sigma,r) &= \sum_{i=-100}^{100}\begin{cases}\text{Gaussian}(x,\sigma) & \text{round}(i,r)=x\\ 0 & \text{round}(i,r)\neq x\end{cases} \\
p(x,\sigma,r) &= \frac{\text{rounded\_Gaussian(x},\sigma,\text{r)}}{\sum\limits_{i=-100}^{100}\text{rounded\_Gaussian}(x,\sigma,r)}
\end{aligned} \tag{VII.2}$$

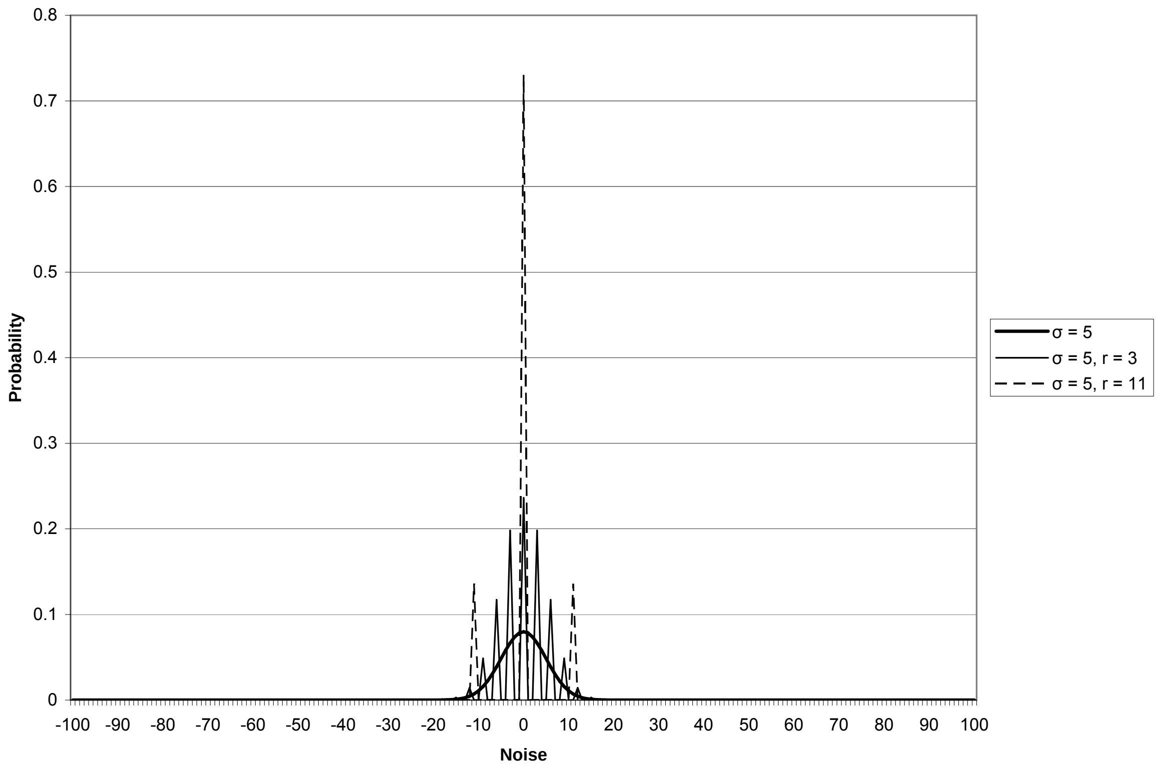


Figure VII.3: Rounded discrete Gaussian $\sigma = 5$ PDFs

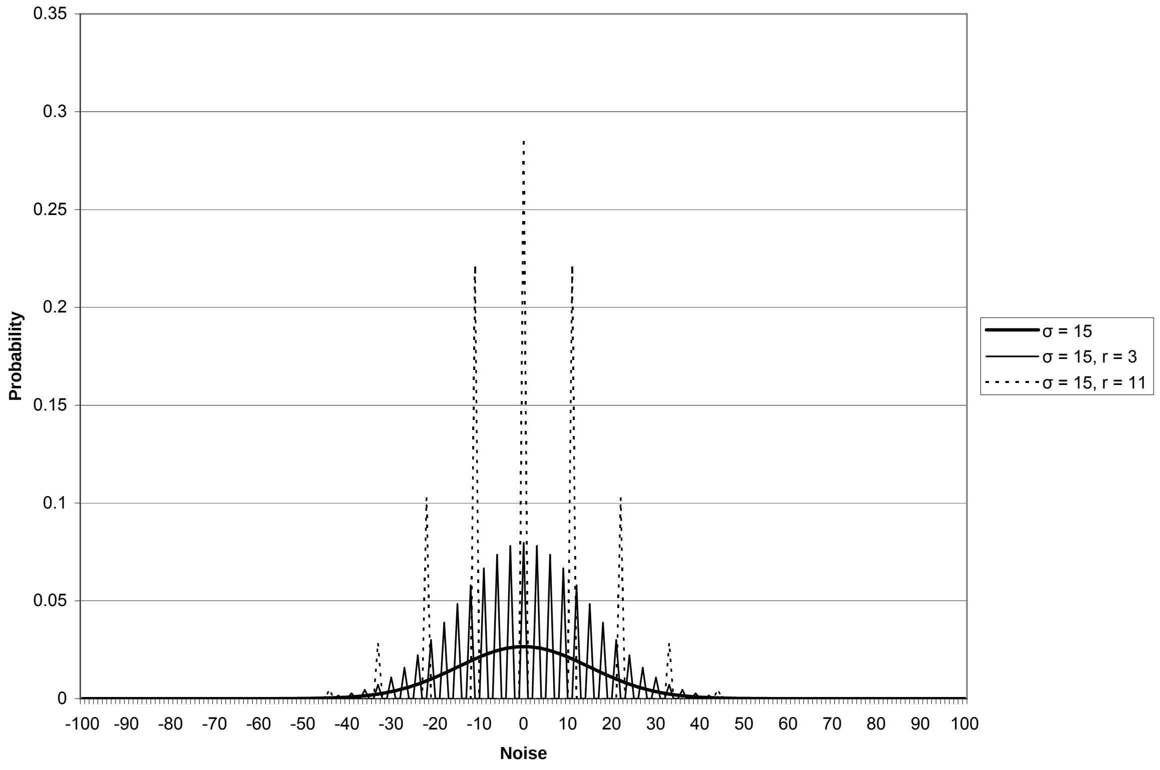


Figure VII.4: Rounded discrete Gaussian $\sigma = 15$ PDFs

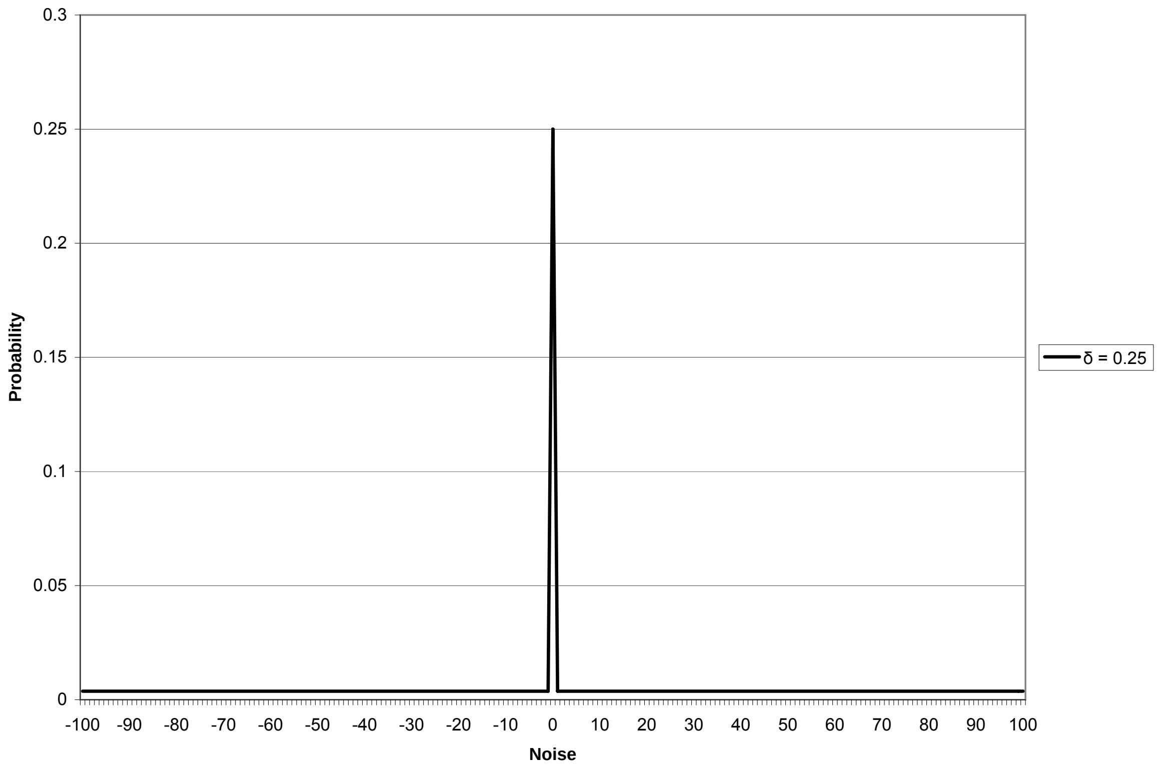


Figure VII.5: Impulse distribution PDFs

Impulse: The tested impulse family of noise distributions incorporate a large probability located at $x = 0$ and all remaining probability is equally distributed throughout the range -100 to 100. Impulse noise distributions are characterized by Equation VII.3. This experiment tested impulse functions with δ values of 0.25, 0.3, and 0.5. Figure VII.5 shows the distribution for $\delta = 0.25$. When $\delta = 1$, the distribution is the same as No Noise.

$$p(x,\delta) = \begin{cases} \delta & x = 0 \\ \\ \frac{1-\delta}{200} & x \neq 0 \end{cases} \tag{VII.3}$$

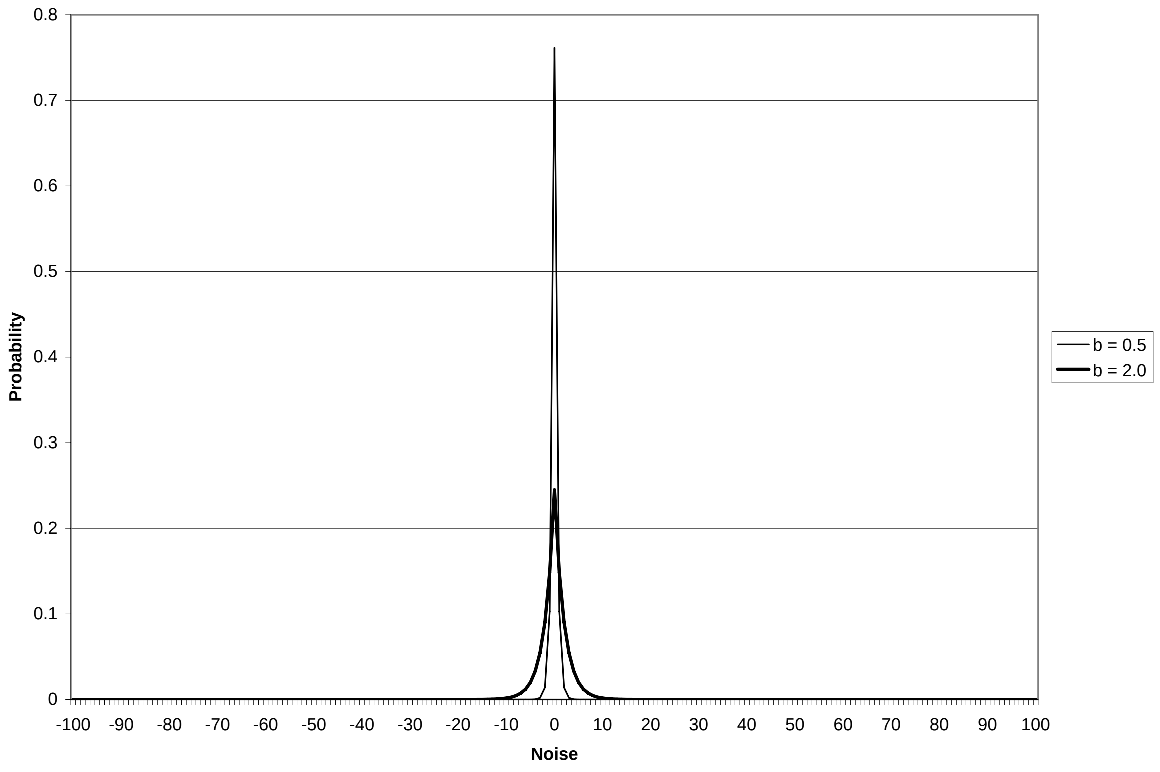


Figure VII.6: Discrete Laplace PDFs

Discrete Laplace: The discrete Laplace family of noise distributions are characterized by Equation VII.4. Figure VII.6 displays Laplace distributions with $b = 0.5$ and $b = 2$. During this experiment, four Laplace distributions were tested, $b = 0.5$, $b = 1$, $b = 1.5$, and $b = 2$.

$$\begin{aligned} \text{Laplace}(x,b) &= \tfrac{1}{2b}e^{\frac{-|x|}{b}} \\ p(x,b) &= \frac{\text{Laplace}(\text{x},\text{b})}{\sum\limits_{i=-100}^{100}\text{Laplace}(\text{i},\text{b})} \end{aligned} \tag{VII.4}$$

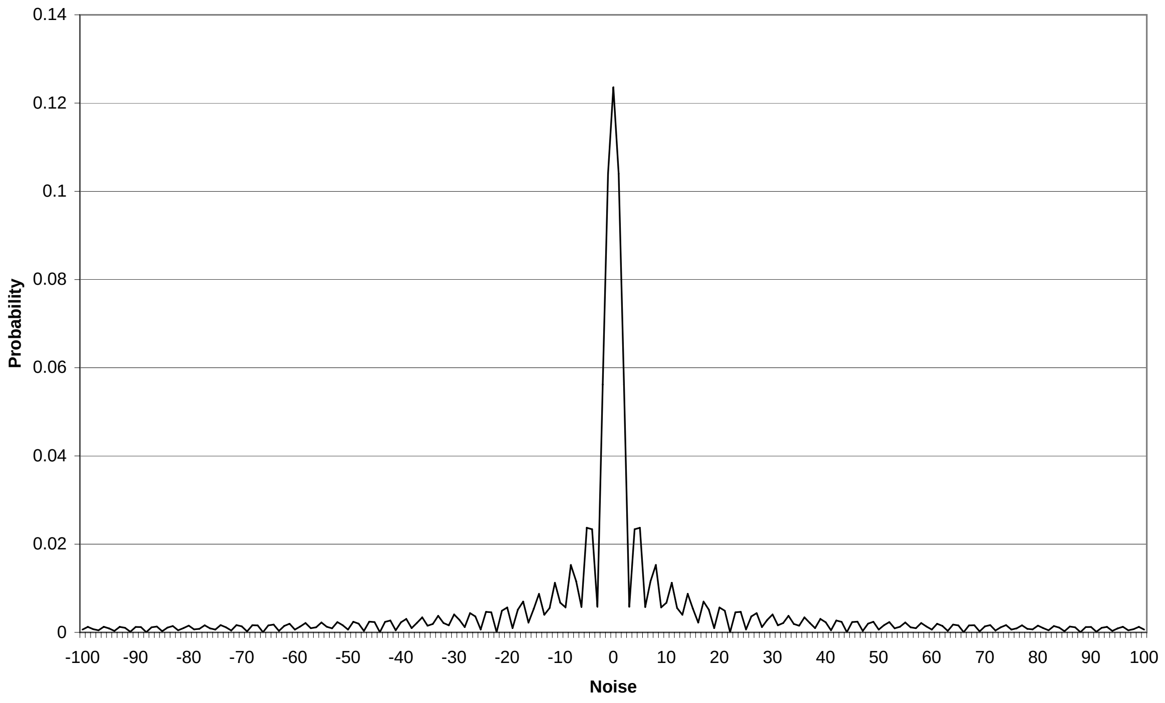


Figure VII.7: PDF for the sinc noise distribution

Sinc: The sinc noise distribution was inspired by the sinc function (Equation VII.5). Equation VII.6 characterizes the sinc noise distribution. Since the sinc noise distribution does not contain any parameters, only one instance was tested. A plot of the sinc noise distribution is provided in Figure VII.7.

$$\operatorname{sinc}(x) = \begin{cases} 1 & x = 0 \\ \frac{\sin(x)}{x} & x \neq 0 \end{cases} \tag{VII.5}$$

$$p(x,b) = \frac{\text{sinc(x)}}{\sum_{i=-100}^{100} \text{sinc(x)}} \tag{VII.6}$$

Table VII.1: Noise distribution effects on average recallable reading counts

Distribution	Parameters	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Gaussian	$\sigma = 1$	1243.81	749.498	772.833	592.845	692.446	1243.76	894.168	629.925
	$\sigma = 1.25$	1335.31	751.433	775.058	594.422	698.224	1335.26	844.204	613.593
	$\sigma = 1.5$	1405.43	752.783	776.408	595.257	699.948	1405.38	798.527	599.905
	$\sigma = 2$	1503.69	754.353	777.978	596.079	698.482	1503.63	747.124	580.019
	$\sigma = 5$	1712.59	757.117	780.742	596.949	682.631	1712.53	677.417	540.814
	$\sigma = 5, r = 3$	1430.29	753.138	776.763	595.562	699.866	1430.24	789.532	595.348
	$\sigma = 5, r = 5$	1250.29	749.668	773.129	592.983	693.783	1250.24	901.63	629.616
	$\sigma = 5, r = 11$	963.138	726.616	745.011	583.947	623.624	963.108	824.275	629.864
	$\sigma = 15$	1789.04	757.794	781.419	596.995	671.039	1788.98	677.542	529.13
	$\sigma = 15, r = 3$	1666.57	756.459	780.084	596.917	685.058	1666.52	714.292	549.129
	$\sigma = 15, r = 5$	1568.62	755.203	778.828	596.578	693.498	1568.57	751.274	567.181
	$\sigma = 15, r = 11$	1338.81	751.379	775.004	594.51	700.785	1338.77	878.958	614.091
Impulse	$\delta = 0.25$	883.529	718.754	737.316	582.242	637.197	883.49	826.471	664.399
	$\delta = 0.3$	917.292	725.406	743.978	584.356	658.369	917.255	865.527	678.752
	$\delta = 0.5$	986.244	737.493	757.01	588.988	702.128	986.206	931.357	702.491
Laplace	$b = 0.5$	956.453	724.642	742.695	583.14	610.775	956.416	807.529	618.877
	$b = 1$	1276.48	749.24	772.826	592.576	676.176	1276.43	882.602	612.133
	$b = 1.5$	1435.65	752.569	776.194	595.138	688.393	1435.59	783.751	588.124
	$b = 2$	1529.15	754.25	777.875	596.033	689.263	1529.1	739.718	571.789
Sinc	———	1708.94	756.715	780.34	596.911	678.045	1708.89	740.522	540.631
Type1	———	1134.94	743.156	765.058	588.315	644.386	1134.91	932.682	617.406
None	———	642.234	603.25	610.641	541.375	536.115	642.21	546.603	556.673
With Impulse	Average	1303.568	740.042	761.69	590.096	670.92	1303.522	797.987	601.359
	Std	311.835	32.819	36.544	11.956	40.018	311.826	94.847	45.574
	Rank	1	5	4	8	6	2	3	7
Without Impulse	Average	1362.707	742.066	764.152	590.87	671.712	1362.66	785.913	588.645
	Std	293.378	34.855	38.729	12.681	41.733	293.369	95.156	33.627
	Rank	1	5	4	7	6	2	3	8

None: The No Noise case was also tested. Equation VII.7 describes how the no noise case was implemented.

$$p(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases} \tag{VII.7}$$

VII.2 The results

The effects of the 21 new noise distributions described in Chapter VII.1 were explored in a simulation evaluation. Each of the 21 new noise distributions, along with the original noise distribution, were individually used to add noise to the readings collected in the signal strength estimation domain. All 64 paths described in Chapters IV and V were used during testing. Except for changing the noise distribution in the environment, the testing conditions were identical to those described in Chapter V.

The noise distribution present in the environment can alter the number of readings filtered by a forgetting algorithm. Table VII.1 presents the average number of recallable readings for the noise distributions when applied to the seven forgetting algorithms and No Forgetting. The left-most column lists each distribution's family (Gaussian, Impulse, Laplace, sinc, Original, or No Noise). The second column lists the parameters that define the individual noise distributions. The Gaussian and rounded Gaussian noise distributions can be differentiated by the presence of the r parameter, as the parameter is specific to the rounded Gaussian class. The remaining columns present the average number of recallable readings. The ranking values were calculated after rounding to minimize the effects of noise. Table VII.1 provides two sets of overall descriptive statistics, labeled "With Impulse" and "Without Impulse". The former includes all of the tested noise distributions, while the latter excludes the Impulse noise distributions. The Impulse noise distributions generated significantly more estimation error than the other tested noise distributions (presented below).

When all of the noise distributions were considered, No Forgetting generated the largest number of recallable readings (1303.568) and the SIMPLE algorithm averaged a nearly identical 1303.522 recallable readings. No Forgetting and the SIMPLE algorithm also generated the largest number of recallable readings during the optimization experiments (Chapter V). The Random forgetting algorithm produced the least number of recallable readings 590.096, while the ActSimple algorithm generated the second smallest number (601.359). The SIMPLE Update algorithm resulted in the third most recallable readings (797.987), while the ACT-R algorithm resulted in the third least number of recallable readings (670.92).

The recallable readings metric for No Forgetting and the SIMPLE algorithm resulted in similar large standard deviations of 311.835 and 311.826, respectively. The standard deviation values for the other forgetting algorithms were considerably smaller. The SIMPLE Update algorithm had a standard deviation of 94.847, but the remaining algorithms' standard deviations were all at or below 45.574. The amount of filtering provided by the Queue Static, Queue Dynamic, Random, ACT-R, and ActSimple forgetting algorithms is comparatively consistent in the presence of varying noise distributions.

When the Impulse noise distributions are not included in the results, the ActSimple algorithm filtered the most number of readings (588.645), while Random forgetting filtered the second most number of readings (590.87). The other rank ordering values did not change. However, the average number of recallable readings increased for all forgetting algorithms and No Forgetting, except SIMPLE Update and ActSimple when the Impulse distribution results were not included.

Changing noise distributions within an environment can also affect estimation error performance. Table VII.2 presents the resultant estimation error for each noise distribution averaged across the forgetting algorithms and No Forgetting. The left two columns identify the individual noise distributions in the same fashion as Table VII.1. The noise distributions did not provide a consistent challenge. The three Impulse distributions resulted in significantly larger estimation error (53.833 for $\delta = 0.25$, 50.606 for $\delta = 0.3$, and 37.032 for $\delta = 0.5$) than any other noise distribution. In the environment, the signal strength ranged from 1 to 100 and before the first reading was collected, the robot guessed a value of 50.5 for each location in the

Table VII.2: Noise distribution average estimation error

Distribution	Parameters	Average	Std
Gaussian	$\sigma = 1$	3.853	1.286
	$\sigma = 1.25$	3.940	1.279
	$\sigma = 1.5$	4.049	1.261
	$\sigma = 2$	4.278	1.230
	$\sigma = 5$	5.485	1.120
	$\sigma = 5, r = 3$	5.377	1.158
	$\sigma = 5, r = 5$	5.337	1.182
	$\sigma = 5, r = 11$	5.231	1.182
	$\sigma = 15$	9.267	1.026
	$\sigma = 15, r = 3$	9.211	1.050
	$\sigma = 15, r = 5$	9.155	1.059
	$\sigma = 15, r = 11$	9.030	1.08
Impulse	$\delta = 0.25$	53.833	2.600
	$\delta = 0.3$	50.606	2.220
	$\delta = 0.5$	37.032	0.437
Laplace	$b = 0.5$	3.882	1.179
	$b = 1$	3.916	1.288
	$b = 1.5$	4.184	1.253
	$b = 2$	4.468	1.220
Sinc	———	11.204	1.785
Original	———	7.250	1.402
None	———	4.194	0.976

environment. This guess results in a maximum average error of 49.5. The average estimation error when $\delta = 0.25$ was 4.333 larger than simply guessing 49.5.

After the sinc distribution's average estimation error of 11.204, the four Gaussian distributions with $\sigma = 15$ generated the largest average error (9.267 for $\sigma = 15$, 9.211 for $\{\sigma = 15, r = 3\}$, 9.155 for $\{\sigma = 15, r = 5\}$, and 9.030 for $\{\sigma = 15, r = 11\}$). Rounding in the rounded Gaussian distributions reduced the number of noise values with a non-zero probability. As the rounding in the $\sigma = 15$ distributions increased, the average estimation error decreased. However, the original noise distribution, which only possesses nine values with a non-zero probability, resulted in the next largest average estimation error (7.250). This error value was larger than the Gaussian distribution with $\sigma = 5$ (5.485) and the Laplace distribution with $b = 2$ (4.468), despite these two distributions having 49 and 53 probability values at or above 0.000001, respectively. Additionally, the Laplace distribution had thirteen values with a probability greater than 0.012 and the Gaussian distribution had twenty one values with a probability larger than 0.01.

The Gaussian distributions with $\sigma = 5$ followed the same trend as the Gaussian distributions with $\sigma = 15$. The distributions' estimation error decreased as rounding increased (5.485 for $\sigma = 5$ to 5.231 for $\{\sigma = 5, r = 11\}$). The Laplace distributions also realized decreasing error as the value of b was reduced (4.468 for $b = 2$ to 3.882 for $b = 0.5$).

The Gaussian distributions with $\sigma = 1.5$, $\sigma = 1.25$, and $\sigma = 1$ generated average estimation errors of 4.049, 3.940, and 3.853, respectively. The No Noise case unintuitively produced greater error (4.194). A small amount of noise appears to have benefited at least one of the forgetting algorithms or No Forgetting.

The individual estimation error values were also recorded and are presented in Table VII.3. The table provides overall descriptive statistics both with and without the Impulse noise distributions. When the Impulse distributions are included, the ActSimple algorithm generated the smallest overall average estimation error (10.755), but the Queue Static algorithm's error was slightly larger (10.837). The SIMPLE Update algorithm, which had the second best estimation error during optimization (on paths 1 - 20) (Chapter V.1), generated an error of 11.502. This value gives SIMPLE Update the fifth smallest estimation error. No Forgetting and the SIMPLE forgetting algorithm tied for the worst average estimation error (12.964). The Queue Static algorithm generated the best estimation error (10.837) out of all three queue-based forgetting algorithms (including Random forgetting).

Several trends exist in the error values for individual noise distributions. During the No Noise case, No Forgetting and the SIMPLE algorithm generated the worst estimation error (5.479). This estimation error value was the smallest achieved by No Forgetting or the SIMPLE algorithm. Every other forgetting algorithm generated less estimation error than No Forgetting and SIMPLE when No Noise was present, but also produced less estimation error than the No Noise case with at least two noise distributions.

Table VII.3: Noise distribution effects on average estimation error

Distribution	Parameters	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Gaussian	$\sigma = 1$	5.77	3.081	2.909	4.283	2.946	5.77	3.472	2.594
	$\sigma = 1.25$	5.853	3.131	2.971	4.361	3	5.853	3.573	2.78
	$\sigma = 1.5$	5.935	3.216	3.08	4.429	3.053	5.935	3.771	2.969
	$\sigma = 2$	6.094	3.386	3.26	4.622	3.219	6.095	4.227	3.323
	$\sigma = 5$	7.056	4.674	4.456	5.816	4.297	7.056	5.768	4.758
	$\sigma = 5, r = 3$	7.05	4.751	4.489	5.857	4.329	7.05	5.233	4.259
	$\sigma = 5, r = 5$	7.033	4.813	4.487	5.885	4.351	7.033	5.05	4.045
	$\sigma = 5, r = 11$	6.805	5.01	4.525	5.805	4.073	6.805	5.213	3.616
	$\sigma = 15$	10.312	8.988	8.441	9.835	7.836	10.312	10.218	8.196
	$\sigma = 15, r = 3$	10.316	9.01	8.41	9.85	7.784	10.316	10.014	7.983
	$\sigma = 15, r = 5$	10.304	9.03	8.405	9.851	7.761	10.304	9.78	7.808
	$\sigma = 15, r = 11$	10.271	9.115	8.332	9.855	7.71	10.271	9.133	7.55
Impulse	$\delta = 0.25$	54.046	50.109	54.474	52.276	57.423	54.045	51.195	57.099
	$\delta = 0.3$	50.722	47.611	50.969	49.063	53.776	50.721	48.476	53.507
	$\delta = 0.5$	37.495	36.392	36.692	36.943	37.446	37.495	36.608	37.184
Laplace	$b = 0.5$	5.604	3.326	3.192	4.287	2.989	5.604	3.575	2.48
	$b = 1$	5.835	3.126	2.977	4.374	2.994	5.835	3.501	2.688
	$b = 1.5$	6.05	3.367	3.202	4.586	3.155	6.05	3.94	3.125
	$b = 2$	6.26	3.616	3.459	4.813	3.363	6.26	4.459	3.513
Sinc	———	12.125	11.041	10.161	11.767	8.911	12.125	14.331	9.17
Original	———	8.794	7.352	6.415	8.213	5.542	8.794	7.7	5.191
None	———	5.479	4.267	4.291	4.333	3.122	5.479	3.813	2.765
With Impulse	Average	12.964	10.837	10.891	11.868	10.867	12.964	11.502	10.755
	Std	14.405	14.173	15.285	14.334	16.189	14.404	14.299	16.138
	Rank	7	2	4	6	3	7	5	1
Without Impulse	Average	7.523	5.49	5.13	6.464	4.76	7.523	6.146	4.674
	Std	2.105	2.659	2.406	2.529	2.113	2.105	3.127	2.266
	Rank	7	4	3	6	2	7	5	1

The Queue Dynamic algorithm experienced the largest difference in estimation error when the No Noise and Gaussian with $\sigma = 1$ distributions were considered. Queue Dynamic generated in an average error of 4.291 with No Noise, but the Gaussian distribution resulted in an estimation error of only 2.909, a difference of 1.382. The Queue Static algorithm experienced a similar trend, as error decreased by 1.186. ActSimple experienced a smaller reduction in error (0.171), but still realized an improvement. These results suggest noise may provide some aid to forgetting algorithms.

The Impulse noise distributions resulted in the largest amount of estimation error for every forgetting algorithm and No Forgetting. While the Impulse noise distribution with $\delta = 0.5$ generated the least amount of error between the Impulse distributions, the error associated with this distribution was over double the next largest estimation error for each forgetting algorithm or No Forgetting. Error values of this magnitude can mask the trends present in the rest of the data. Table VII.3 includes the overall average error for each forgetting algorithm and No Forgetting excluding the Impulse distributions. Without the Impulse distributions, the average estimation error values for the forgetting algorithm and No Forgetting possess smaller magnitudes. ActSimple generated an error of 4.674 instead of 10.755, SIMPLE Update's error changed from 11.502 to 6.146, ACT-R's error dropped from 10.867 to 4.76, and No Forgetting's error improved from 12.964 to 7.523.

Table VII.4: Noise distribution effects on estimation error rankings

Distribution	Parameters	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Gaussian	$\sigma = 1$	7	4	2	6	3	7	5	1
	$\sigma = 1.25$	7	4	2	6	3	7	5	1
	$\sigma = 1.5$	7	4	3	6	2	7	5	1
	$\sigma = 2$	7	4	2	6	1	8	5	3
	$\sigma = 5$	7	3	2	6	1	7	5	4
	$\sigma = 5, r = 3$	7	4	3	6	2	7	5	1
	$\sigma = 5, r = 5$	7	4	3	6	2	7	5	1
	$\sigma = 5, r = 11$	7	4	3	6	2	7	5	1
	$\sigma = 15$	7	4	3	5	1	7	6	2
	$\sigma = 15, r = 3$	7	4	3	5	1	7	6	2
	$\sigma = 15, r = 5$	7	4	3	6	1	7	5	2
	$\sigma = 15, r = 11$	7	4	3	6	2	7	5	1
Impulse	$\delta = 0.25$	5	1	6	3	8	4	2	7
	$\delta = 0.3$	5	1	6	3	8	4	2	7
	$\delta = 0.5$	7	1	3	4	6	7	2	5
Laplace	$b = 0.5$	7	4	3	6	2	7	5	1
	$b = 1$	7	4	2	6	3	7	5	1
	$b = 1.5$	7	4	3	6	2	7	5	1
	$b = 2$	7	4	2	6	1	7	5	3
sinc	———	6	4	3	5	1	6	8	2
Original	———	7	4	3	6	2	7	5	1
None	———	7	4	5	6	2	7	3	1
With Impulse	Average	6.773	3.545	3.091	5.5	2.545	6.727	4.727	2.227
	Std	0.612	1.057	1.151	0.964	2.087	0.935	1.386	1.901
	Rank	8	4	3	6	2	7	5	1
Without Impulse	Average	6.947	3.947	2.789	5.842	1.789	7	5.158	1.579
	Std	0.229	0.229	0.713	0.375	0.713	0.333	0.898	0.902
	Rank	7	4	3	6	2	8	5	1

However, the rank ordering only changed for three forgetting algorithms. The ActSimple algorithm continued to generate the least amount of error, while Queue Static moved from second to fourth. Queue Dynamic switched from fourth to third, and ACT-R improved from third to second. The relative performance of No Forgetting and the SIMPLE and Random forgetting algorithms did not change.

The relative estimation error performance of the forgetting algorithms and No Forgetting was ranked for each noise distribution. These ranking values are presented in Table VII.4. Ranking was performed after rounding to minimize the effects of noise. The overall average rankings were similar for both the With Impulse and the Without Impulse results. All forgetting algorithms maintained the same ranking values except for SIMPLE, which switched from a tie with No Forgetting (With Impulse) to eighth place. The performance deviations between No Forgetting and SIMPLE were very small (Table VII.3) and this change in rank ordering does not appear to represent a large change in performance. The overall rank ordering in Table VII.4 is identical to the Without Impulse overall estimation error ranks in Table VII.3, except for SIMPLE.

The ActSimple and ACT-R algorithms generated the best overall rank values, but generated the worst performance on the two Impulse distributions with $\delta = 0.25$ and $\delta = 0.3$. Additionally, the Queue Dynamic algorithm resulted in the third best overall rank value, but the sixth largest estimation error when tested with the two Impulse noise distributions. The SIMPLE algorithm and No Forgetting generated the fourth and fifth best estimation error values.

The statistical significance of the relative estimation error performance results from Table VII.3 was evaluated with four single-sided Wilcoxon signed rank tests and the results are presented in Table VII.5. The first two tests, labeled *With Impulse - Less than No Forgetting* and *With Impulse - ActSimple less than*, evaluated the statistical significance of the results when the Impulse distributions were included. The last two tests, labeled *Without Impulse - Less than No Forgetting* and *Without Impulse - ActSimple less than*, evaluated the statistical significance when the Impulse distributions were not included. The first and third tests evaluated if the forgetting algorithms listed in the left-most column resulted in less average estimation error than No Forgetting. The second and fourth tests evaluated if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms. The α value for these tests was set to 0.05. Results that were found to be significant are indicated in bold.

Table VII.5: Comparison of estimation error

	With Impulse						Without Impule					
	Less than No Forgetting			ActSimple less than			Less than No Forgetting			ActSimple less than		
	n	v	p	n	v	p	n	v	p	n	v	p
No Forgetting	——	——	——	22	26.0	**0.0003**	——	——	——	19	0.0	<**0.0001**
Queue Static	22	0.0	<**0.0001**	22	56.0	**0.0104**	19	0.0	<**0.0001**	19	2.0	<**0.0001**
Queue Dyanmic	22	3.0	<**0.0001**	22	68.0	**0.0293**	19	0.0	<**0.0001**	19	12.0	**0.0001**
Random	22	0.0	<**0.0001**	22	44.0	**0.0030**	19	0.0	<**0.0001**	19	0.0	<**0.0001**
ACT-R	22	41.0	**0.0021**	22	71.0	**0.0369**	19	0.0	<**0.0001**	19	65.0	0.1206
SIMPLE	21	145.0	0.8462	22	26.0	**0.0003**	18	145.0	0.9966	19	0.0	<**0.0001**
SIMPLE Update	22	17.0	<**0.0001**	22	43.0	**0.0026**	19	16.0	**0.0003**	19	0.0	<**0.0001**
ActSimple	22	26.0	**0.0003**	——	——	——	19	0.0	<**0.0001**	——	——	——

Results from the first test (column *With Impulse - Less than No Forgetting*) were significant for Queue Static ($v = 0.0$, $p < 0.0001$), Queue Dynamic ($v = 3.0$, $p < 0.0001$), Random ($v = 0.0$, $p < 0.0001$), ACT-R ($v = 41.0$, $p = 0.0021$), SIMPLE Update ($v = 17.0$, $p < 0.0001$), and ActSimple ($v = 26.0$, $p = 0.0003$). Results from SIMPLE were not significant ($v = 145.0$, $p = 0.8462$). These results suggest that the results presented in Chapter V were not dependent on the original tested noise distribution. Algorithms based on Human-Inspired Forgetting may be applicable to a wide range of environmental conditions. Additionally, the forgetting algorithms' parameterizations may not require constant re-optimization as real world environments and domains evolve over time.

Each of the comparisons in the second test (column *With Impulse - ActSimple less than*) were significant.

No Forgetting ($v = 26.0,\ p = 0.0003$), Queue Static ($v = 56.0,\ p = 0.0104$), Queue Dynamic ($v = 68.0,$ $p = 0.0293$), Random($v = 44.0$, $p = 0.0030$), ACT-R ($v = 71.0$, $p = 0.0369$), SIMPLE ($v = 26.0$, $p = 0.0003$), and SIMPLE Update ($v = 43.0,\ p = 0.0026$) each resulted in less estimation error than ActSimple. These results suggest that the relative benefits of ActSimple may be resilient to some changes in environmental conditions and the results presented in Chapter V were not a result of a particular set of tested conditions.

The Impulse distributions resulted in substantially more estimation error than the other tested noise distributions and their inclusion in the results had the potential to alter the relative performance of the forgetting algorithms. The *Without Impulse* Wilcoxon signed rank tests evaluated if the benefits of ActSimple and Human-Inspired Forgetting were still present when the Impulse noise distributions were removed. The *Without Impulse - Less than No Forgetting*) test evaluated if the forgetting algorithms generated less estimation error than No Forgetting and the results were significant for Queue Static ($v = 0.0$, $p < 0.0001$), Queue Dynamic ($v = 0.0$, $p < 0.0001$), Random ($v = 0.0$, $p < 0.0001$), ACT-R ($v = 0.0$, $p < 0.0001$), SIMPLE Update ($v = 16.0,\ p = 0.0003$), and ActSimple ($v = 0.0,\ p < 0.0001$). Results from SIMPLE were not significant ($v = 145.0,\ p = 0.9966$). When the Impulse noise distributions were removed, the forgetting algorithms continued to generate less estimation error than No Forgetting.

The fourth Wilcoxon signed rank test evaluated if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms when the Impulse distributions were not included (column *Without Impulse - ActSimple less than*). Results from No Forgetting ($v = 0.0,\ p < 0.0001$), Queue Static ($v = 2.0,$ $p < 0.0001$), Queue Dynamic ($v = 12.0$, $p = 0.0001$), Random ($v = 0.0$, $p < 0.0001$), SIMPLE ($v = 0.0$, $p <$ 0.0001), and SIMPLE Update ($v = 0.0,\ p < 0.0001$) were significant, while results from ACT-R ($v = 65.0,$ $p = 0.1206$) were not significant. Results from this test suggest the ability for ActSimple to generate less estimation error than No Forgetting and the forgetting algorithms that do not include trace-based decay was not a result of the Impulse distributions. The difference in estimation error between ActSimple and ACT-R was not significant, suggesting trace-based decay may be an effective method to improve system accuracy, even in the presence of diverse environmental conditions.

Individual paths can influence the amount of estimation error that is generated. Some paths are more challenging than others and individual forgetting algorithms may respond differently depending on path properties. Table VII.6 presents the estimation error results from Table VII.3 for only paths 1 - 20, instead of the full set of 64 paths. This table is formatted identical to Table VII.3. The forgetting algorithms and No Forgetting each generated less overall average estimation error (With Impulse) on the full set of noise distributions when only paths 1 - 20 were included. Estimation error for the Without Impulse results followed a different trend. Excluding the SIMPLE Update algorithm, the forgetting algorithms and No Forgetting each averaged more error with paths 1 - 20. The Impulse noise distribution had a smaller negative impact on the original set of

Table VII.6: Noise distribution effects on average estimation error (paths 1 - 20)

Distribution	Parameters	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Gaussian	$\sigma = 1$	6.017	3.053	2.856	4.699	3.422	6.017	2.404	2.653
	$\sigma = 1.25$	6.101	3.335	3.132	4.847	3.538	6.101	2.77	2.965
	$\sigma = 1.5$	6.183	3.609	3.432	4.933	3.637	6.183	3.168	3.28
	$\sigma = 2$	6.345	3.925	3.839	5.166	3.877	6.345	3.916	3.855
	$\sigma = 5$	7.309	5.427	5.31	6.496	5.279	7.308	5.904	5.964
	$\sigma 5, r = 3$	7.311	5.269	5.054	6.447	5.151	7.311	4.783	4.86
	$\sigma = 5, r = 5$	7.305	4.939	4.623	6.401	5.059	7.305	4.045	4.369
	$\sigma = 5, r = 11$	7.13	4.503	4.311	6.125	4.517	7.13	3.628	3.503
	$\sigma = 15$	10.54	9.955	9.641	10.665	9.463	10.54	10.279	10.099
	$\sigma = 15, r = 3$	10.547	9.892	9.562	10.637	9.282	10.547	10.029	9.58
	$\sigma = 15, r = 5$	10.541	9.86	9.488	10.636	9.181	10.541	9.663	9.15
	$\sigma = 15, r = 11$	10.526	9.587	8.993	10.554	8.926	10.526	8.384	8.479
Impulse	$\delta = 0.25$	41.226	37.318	41.81	39.687	43.759	41.225	39.509	43.626
	$\delta = 0.3$	38.783	35.295	39.217	37.362	41.062	38.782	37.323	40.927
	$\delta = 0.5$	29.075	27.627	28.18	28.566	29.036	29.074	27.745	28.513
Laplace	$b = 0.5$	5.852	2.608	2.748	4.462	3.262	5.852	2.114	2.193
	$b = 1$	6.091	3.154	2.949	4.804	3.469	6.091	2.511	2.752
	$b = 1.5$	6.31	3.806	3.574	5.129	3.762	6.31	3.504	3.499
	$b = 2$	6.524	4.173	4.064	5.39	4.07	6.524	4.281	4.13
sinc	———	12.391	11.8	11.273	12.449	10.414	12.391	13.26	10.752
Original	———	9.164	7.394	6.322	8.72	6.278	9.164	6.076	5.441
No Noise	———	5.714	3.439	3.513	4.228	3.221	5.714	2.281	2.214
With Impulse	Average	11.681	9.544	9.722	10.836	9.985	11.681	9.435	9.673
	Std	10.416	10.21	11.379	10.376	11.862	10.416	10.956	11.964
	Rank	7	2	4	6	5	7	1	3
Without Impulse	Average	7.784	5.775	5.510	6.989	5.569	7.784	5.421	5.249
	Std	2.106	2.958	2.805	2.682	2.527	2.106	3.308	2.883
	Rank	8	5	3	6	4	7	2	1

paths (1 - 20).

The ActSimple algorithm only generated the third best estimation error (9.673) with paths 1 - 20 when the Impulse distributions were included. SIMPLE Update realized the least amount of estimation error (9.435) and Queue Static averaged the second least average error (9.544). The ACT-R algorithm produced the fifth least amount of error (9.985). When the Impulse distributions were not included, ActSimple averaged the least amount of error (5.249) and SIMPLE Update generated the second least amount of error (5.421). Queue Static's relative performance dropped to fifth place with an error of 5.775.

The SIMPLE Update algorithm generated the best estimation error when the Impulse noise distributions were included in the results and the second best error when the Impulse noise distributions were not included. However, SIMPLE Update only averaged the fifth best estimation error when processing results from all 64 paths (Table VII.3). Some of the paths from path 21 - path 64 pose a greater challenge to SIMPLE Update than some other forgetting algorithms.

Rank ordering of the estimation error resulting from paths 1 - 20 was calculated and is presented in Table VII.7. Ranking was performed after rounding to minimize the effects of noise. The overall rank ordering averages changed from the ranking of overall estimation error for paths 1 - 20 when the Impulse

Table VII.7: Noise distribution effects on estimation error rankings (paths 1 - 20)

Distribution	Parameters	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Gaussian	$\sigma = 1$	7	4	3	6	5	7	1	2
	$\sigma = 1.25$	7	4	3	6	5	7	1	2
	$\sigma = 1.5$	7	4	3	6	5	7	1	2
	$\sigma = 2$	7	5	1	6	3	7	4	2
	$\sigma = 5$	8	3	2	6	1	7	4	5
	$\sigma = 5, r = 3$	7	5	3	6	4	7	1	2
	$\sigma = 5, r = 5$	7	4	3	6	5	7	1	2
	$\sigma = 5, r = 11$	7	4	3	6	5	7	2	1
	$\sigma = 15$	6	3	2	8	1	6	5	4
	$\sigma = 15, r = 3$	6	4	2	8	1	6	5	3
	$\sigma = 15, r = 5$	6	5	3	8	2	6	4	1
	$\sigma = 15, r = 11$	6	5	4	8	3	6	1	2
Impulse	$\delta = 0.25$	5	1	6	3	8	4	2	7
	$\delta = 0.3$	5	1	6	3	8	4	2	7
	$\delta = 0.5$	8	1	3	5	6	7	2	4
Laplace	$b = 0.5$	7	3	4	6	5	7	1	2
	$b = 1$	7	4	3	6	5	7	1	2
	$b = 1.5$	7	5	3	6	4	7	2	1
	$b = 2$	7	4	1	6	2	7	5	3
sinc	———	5	4	3	7	1	5	8	2
Original	———	7	5	4	6	3	7	2	1
No Noise	———	7	4	5	6	3	7	2	1
With Impulse	Average	6.636	3.727	3.182	6.091	3.862	6.455	2.591	2.636
	Std	0.848	1.279	1.296	1.306	2.077	0.963	1.894	1.761
	Rank	8	4	3	6	5	7	1	2
Without Impulse	Average	6.737	4.158	2.895	6.474	3.316	6.684	2.684	2.105
	Std	0.653	0.688	0.994	0.841	1.600	0.582	2.029	1.049
	Rank	8	5	3	6	4	7	2	1

distributions were included. ActSimple improved from the third least estimation error to the second smallest average rank. The SIMPLE Update algorithm maintained the best performance. Queue Static dropped from second to fourth and Queue Dynamic moved from fourth to third. The ACT-R algorithm remained fifth. Rank positions, when the Impulse distributions were not included, did not change from Table VII.6 to Table VII.7.

The rank ordering of performance in Tables VII.3 and VII.6 was identical for the Impulse distributions with $\delta = 0.25$ and $\delta = 0.3$. ActSimple and ACT-R generated more error than the other forgetting algorithms and No Forgetting with these noise distributions for both the full set of paths (paths 1 - 64) and the smaller original set (paths 1 - 20).

VII.3 Summary

Noise in the environment can influence the effectiveness of a forgetting algorithm. This chapter presented an experiment where the performance of No Forgetting and the seven forgetting algorithms was tested with twenty one new noise distributions. Results from this experiment demonstrate the stability and flexibility of the benefits afforded by Human-Inspired Forgetting.

No Forgetting did not generate the largest amount of estimation error for each of the new noise distributions, but tied for the greatest amount of overall average estimation error and never resulted in better than the fifth best estimation error value. The Human-Inspired Forgetting algorithms managed to generate less estimation error, despite the changes to the noise distribution present in the environment. In many real-world environments, accurately modeling the noise and other environmental properties that can influence a mobile robot's performance is very challenging, if not impossible. The results from this chapter suggest that if Human-Inspired Forgetting is utilized in mobile robots operating in real-world environments, the algorithms may increase system accuracy without requiring constant re-optimization.

The ActSimple algorithm averaged the least amount of average estimation error and appears to be an effective and reliable means of incorporating Human-Inspired Forgetting into robotic systems. The algorithm did not generate the smallest amount of estimation error for each tested noise distribution, but resulted in the smallest average estimation error and the best average rank value. Forgetting algorithms may not be able to guarantee the best possible performance under every condition, but ActSimple appears to be an effective means of improving average performance.

Algorithmic complexity can be an important factor when selecting and utilizing an algorithm. The estimation error values from Table VII.3 were contrasted with each algorithm's number of parameters and is presented in Figure VII.8. Each algorithm is represented with a unique symbol and error values including the Impulse noise distributions are shown in black, while the error values not including the Impulse distributions are shown in gray.

No Forgetting does not possess any parameters and tied for the largest amount of estimation error. ActSimple utilizes the largest number of parameters and resulted in the smallest amount of estimation error. However, a direct relationship between parameter count and estimation error is not shown. SIMPLE and SIMPLE Update each possess seven parameters, yet both resulted in more estimation error than ACT-R, Queue Static, and Queue Dynamic. Other factors appear to have an impact on the effectiveness of a forgetting algorithm to improve system accuracy. Additionally, ActSimple was optimized with only the original noise distribution, but still generated the smallest amount of estimation error. The additional parameters in ActSimple do not appear to cause over-learning, but may aid in ActSimple's ability to perform well across multiple noise distributions.

Human-Inspired Forgetting may improve system accuracy, but the filtering performed by the forgetting algorithms may also reduce the number of data points requiring potentially computationally expensive processing by existing robotic algorithms. A comparison between the forgetting algorithms resultant estimation error (Table VII.3) and the resultant number of recallable readings (Table VII.1) is presented in Figure VII.9. This figure uses the same key as Figure VII.8.

The Impulse noise distributions substantially increased the average estimation error generated by No

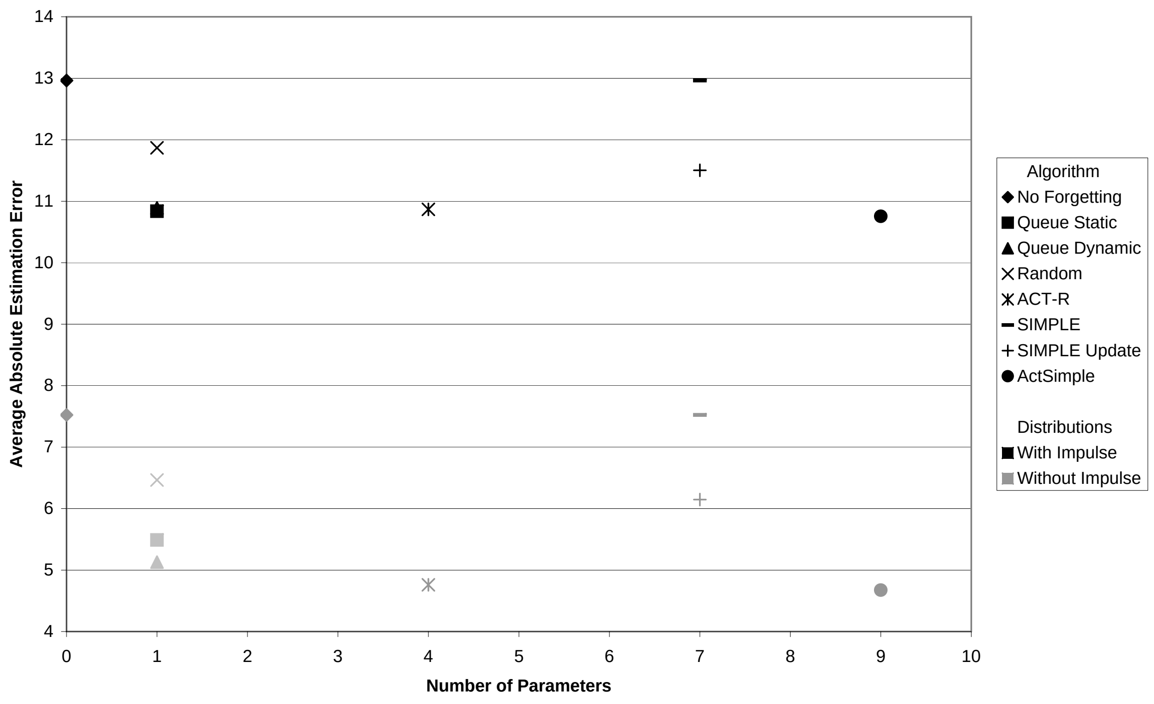


Figure VII.8: Number of parameters vs. average absolute estimation error

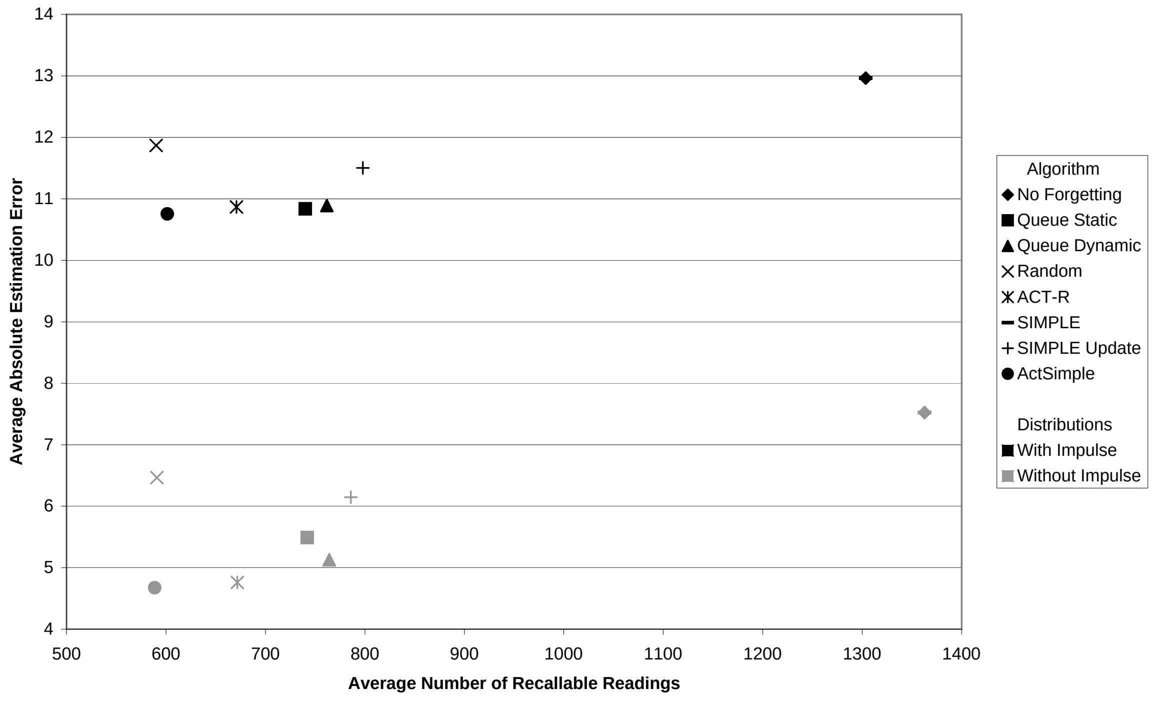


Figure VII.9: Average number of recallable readings vs. average estimation error

Forgetting and the forgetting algorithms, but the relative number of recallable readings and estimation error values are similar. No Forgetting and SIMPLE resulted in the most estimation error, but also the largest number of recallable readings. This data suggests that in noisy environments, the quantity of data available to an existing robotic algorithm may not result in improved accuracy. ActSimple generated the least amount of estimation error and resulted in at most the second smallest number of recallable readings. The ability to intelligently select a subset of data collected from the environment may have a greater influence on the effectiveness of a mobile robot to improve system accuracy.

ActSimple is the most complex forgetting algorithm that was tested, possessing the largest number of parameters, but was able to average the smallest amount of estimation error and at most the second smallest number of recallable readings. Robotic systems that operate in dynamic real-world environments and employ computationally expensive existing robotic algorithms may realize improved accuracy, while decreasing overall computational load with the use of the ActSimple forgetting algorithm. Generating the best average estimation error over the full set of tested noise distributions, the ActSimple algorithm may also not require constant re-optimization in order to provide benefits to the system.

CHAPTER VIII

Evaluating the Performance of Forgetting at Each Point Along the Paths

During the optimization experiment (Chapter V), the parameters for each forgetting algorithm were optimized by minimizing the resultant WiFi signal strength estimation error generated at the end of each path. Neither the absolute nor the relative performance of a forgetting algorithm parameterization will necessarily be constant throughout a path. At the beginning of a path, most forgetting algorithms and parameterizations generate nearly identical estimation values; however, as the robot continues along the path, the performance across the forgetting algorithms may diverge. This chapter describes an experiment in which forgetting algorithm performance data was gathered throughout the entire path, not just at the end. During this experiment, item recalls did not affect a reading's activation level and only the effects of perception count were included.

This experiment used paths 1 - 20 (Chapter IV.4) and the same simulation conditions used during optimization (Chapter V). The collected results were averaged over 100 instances of each path. These path instances were not the same as the path instances used to optimize the forgetting algorithms. The resultant estimation values for each path were plotted. The most interesting results are provided in this chapter and the remaining results are provided in Appendix A. Performance along paths 1, 2, 4, 5, 8, 9, 10, 12, 13, 17, and 20 are described in detail in this chapter.

VIII.1 Performance Along Individual Paths

The performance along all paths exhibited a few general trends. Path 2, provided in Figure VIII.1, is representative of these trends. At the beginning of each path, no WiFi readings were available and estimation error was significant. While the initial estimation error was dependent on a path's initial basestation configuration, the forgetting algorithms and No Forgetting each generated identical error values. As the robot began collecting WiFi signal strength readings, the estimation error generally began to decrease, although the decrement rate depended on the path. After approximately 500 - 650 time steps (roughly 600 steps for Path 2), the performance of the forgetting algorithms began to diverge on most paths. The degree that the individual forgetting algorithms' performance diverged was path dependent. None of the conditions generated monotonically decreasing error, even when only paths containing a single basestation configuration were analyzed.

VIII.1.1 Performance of Forgetting when Applied to Path 2

The benefits of the individual forgetting algorithms varied across paths, but also experienced sizable variability along each path. Figure VIII.1 presents the along the path results for Path 2. This path involved one complete cycle through the simulation environment using the four basestation configuration. At the beginning of this path, no readings were available and the estimation error was very large (42.0696). After the first reading was collected, the average error dropped sharply to 30.0406, as is visible in Figure VIII.1. The estimation error continued to decline until 43 readings were collected. At this point, the error began to increase for approximately 138 readings before returning to a declining trend. Around the 600th time step, the estimation error generated by the different forgetting algorithms began to diverge. The SIMPLE Update and ACT-R algorithms both experienced a larger increase in estimation error than the other forgetting algorithms or No Forgetting. During the period of increased error, from approximately reading 700 to 1150, ACT-R reached an error value of 13.759 and SIMPLE Update's error increased to 13.298. The error generated by ACT-R and SIMPLE Update at the end of the path had returned to levels similar to the other forgetting algorithms and No Forgetting. The trends shown in path 2 are representative of the performance generated along path 3 (Figure A.1).

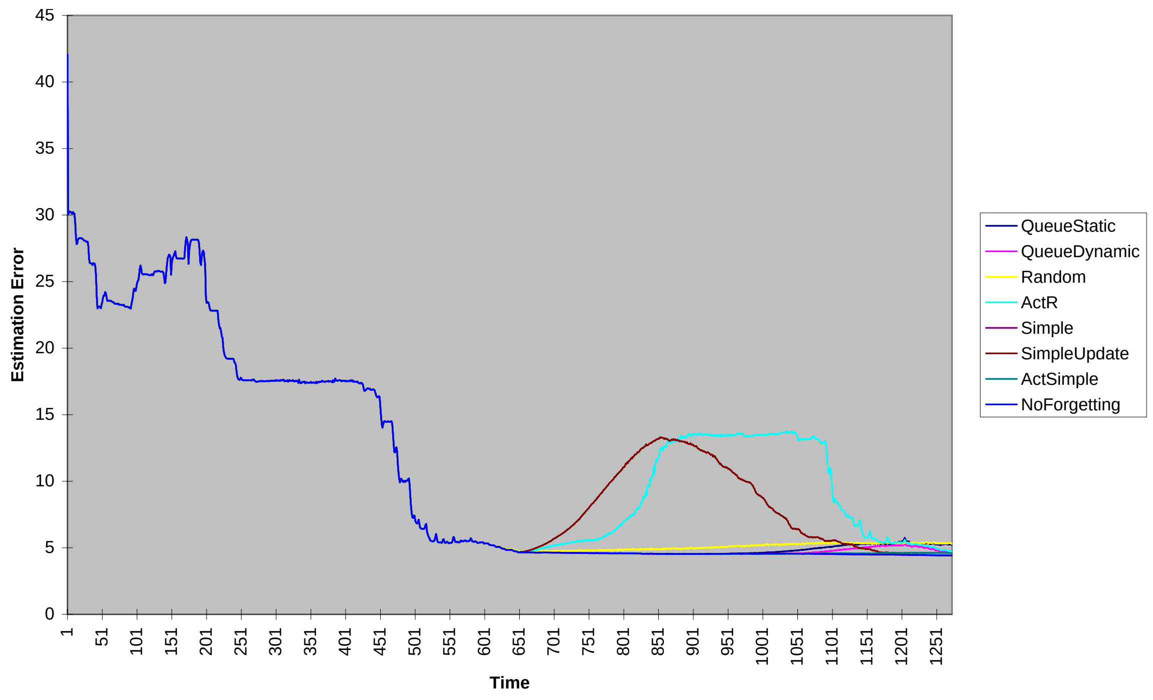
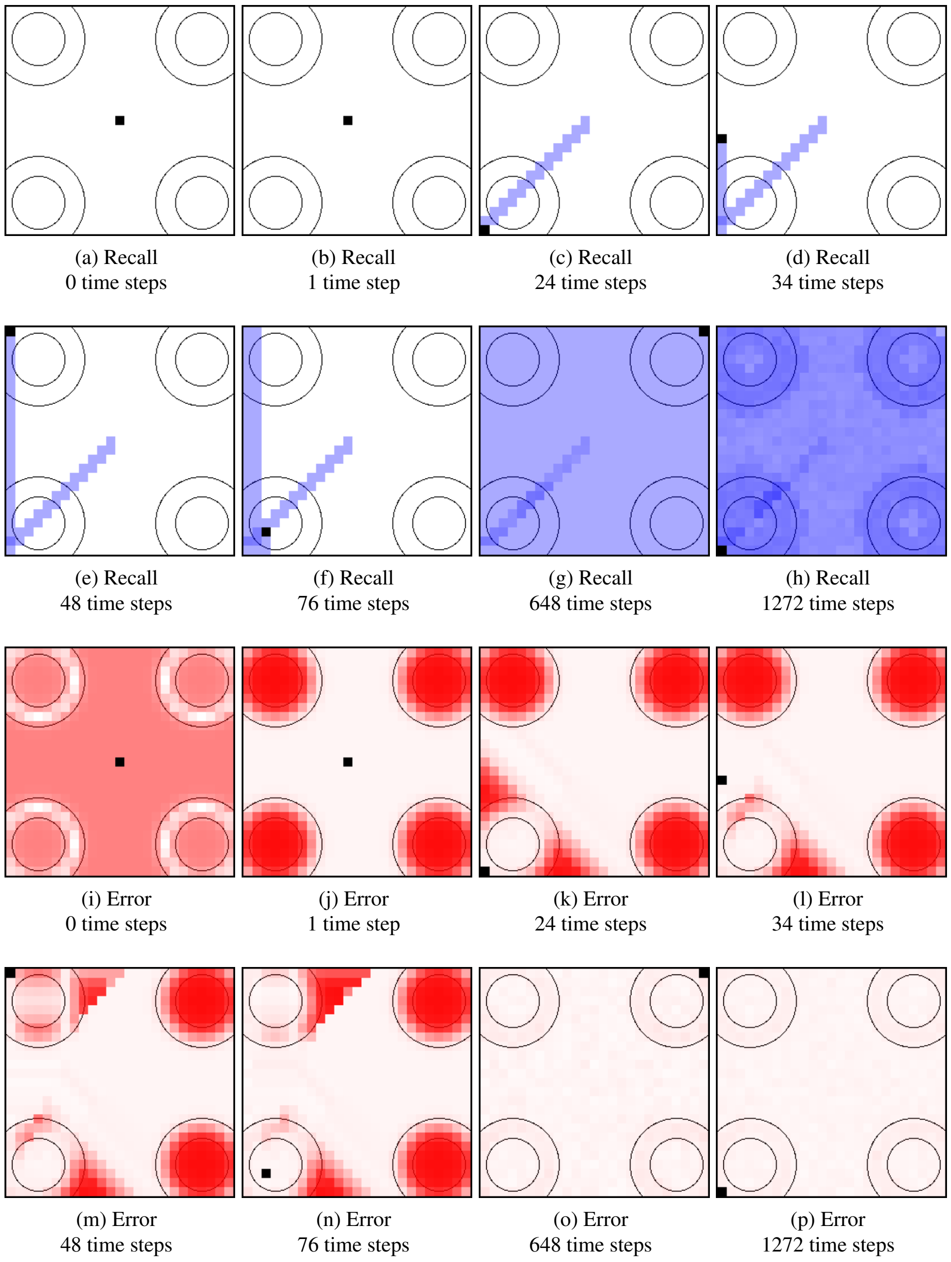


Figure VIII.1: Along path results for path 2 - 1 cycle 4 basestations

Performance of the forgetting algorithms and No Forgetting at each point in time along the path was influenced by the environment. Estimation error values and the number of recallable readings were often inconsistent at any given point in time. Figure VIII.2 graphically presents the individual error and recallable reading counts throughout the environment at selected points in time when No Forgetting was present. Figures VIII.2a - VIII.2h show the average number of recallable readings at each location in the environment. The recall maps indicate recallable readings with blue. Darker colors represent larger numbers of recallable readings. Recall values within these maps are clipped to a value of three readings to allow for discrimination of values between 0 to 3. The blue color does not directly present the probability of recalling an item, simply the average number of recallable items. As a result, two harder to recall readings may be represented with a darker color than one easier to recall reading. The small black square represents the robot. The robot moves and then takes a reading, thus, the last collected reading is always concealed by the robot indicator. The basestations are represented as two circles in Figure VIII.2. The basestations are located in each corner. The inner circle shows the 90% strength mark for the basestation, while the outer circle shows the radius where the basestation's strength has been reduced to 10%. These circles indicate the transition area for a basestation, where available signal strength shifts from high levels to low. Figures VIII.2i - VIII.2p show the individual error values throughout the environment. Red indicates error, with darker colors representing greater amounts of error. The robot indicator and the basestation markings are the same in the error maps as they are in the recall maps. The full set of error and recall maps for No Forgetting, Queue Static, ACT-R, ActSimple, and SIMPLE Update for all twenty paths are available in Appendix A.

Path 2 is one of the simplest paths tested. The robot started at the center of the environment and traveled to the lower left corner of the environment. The robot proceeded to exhaustively explore the environment traveling from the left side of the environment to the right. At the completion of the trip to the right side of the environment, the robot returned to the lower left corner of the environment, again with an exhaustive search strategy. At least one reading was collected at every location in the environment.

At the beginning of the path, the robot is located in the center of the environment and no readings are available (Figure VIII.2a). The average number of recallable readings is zero for every location in the environment and the associated recall map (Figure VIII.2a) contains no blue markings. The robot does not not have any signal strength readings and is forced to guess a signal strength value of 50.5 for each location (Figure VIII.2i). Substantial error is generated at most locations in the environment, shown in Figure VIII.2i by the large amount of red. The only areas where large amounts of error are not present is in the transition areas for each basestation, identified with each basestation's pair of circles. At this point, the robot made a movement towards the lower left corner of the environment and gathered its first reading (Figure VIII.2b). The robot changed its error estimation for every location in the environment based on this single reading and the resultant error changed significantly (Figure VIII.2j). The robot traveled to the lower left corner of the


(a) Recall
0 time steps

(b) Recall
1 time step

(c) Recall
24 time steps

(d) Recall
34 time steps

(e) Recall
48 time steps

(f) Recall
76 time steps

(g) Recall
648 time steps

(h) Recall
1272 time steps

(i) Error
0 time steps

(j) Error
1 time step

(k) Error
24 time steps

(l) Error
34 time steps

(m) Error
48 time steps

(n) Error
76 time steps

(o) Error
648 time steps

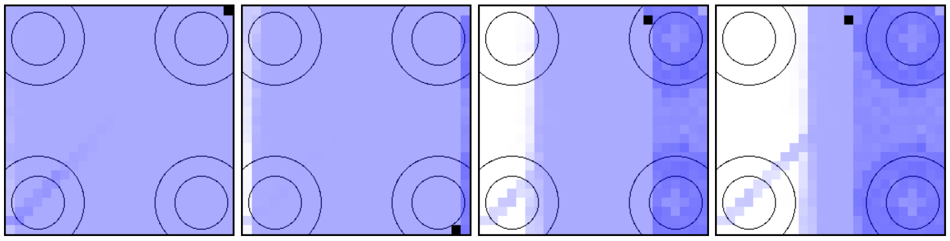
(p) Error
1272 time steps

Figure VIII.2: Path 2 - 1 cycle 4 basestations with No Forgetting error and recall maps

environment collecting signal strength readings (Figure VIII.2c). Signal strength in the center of the environment was at a constant minimum level. Until the robot reached the basestation in the lower left corner, collecting additional readings did not result in visible changes to the robot's estimation error. When the robot reached the lower left basestation and began collecting higher valued strength readings (Figure VIII.2c), the corresponding estimation error began to decrease (Figure VIII.2k). The robot collected additional readings (Figures VIII.2d - VIII.2f), which slowly reduced estimation error near the basestations (Figures VIII.2l - VIII.2n). After the 648th time step (Figure VIII.2g), at least one reading had been collected for every location in the environment and pockets of significant error were removed (Figure VIII.2o). The robot collected additional readings, while returning to the lower left corner of the environment (Figure VIII.2h), but no significant visual change occurred to the error map (Figure VIII.2p).

Environmental conditions had a large impact on the average number of recallable readings at each location. Figure VIII.2h shows the effects of a basestation on the number of recallable readings. Both the center of the environment and the center of the basestations possess an extreme signal strength level (either the minimum or the maximum detectable value). The noise present in the environment is symmetric, but is then clipped, converting the noise into an asymmetric noise source. At the center of the environment, any signal strength reading that would have been negative (due to noise before clipping) is clipped to the minimum detectable signal strength value. Likewise, readings centered on a basestation will be clipped to the maximum detectable signal strength value. The clipping process at these two locations effectively limits the number of unique signal strength readings that can be recorded by the robot. Conversely, the transition areas between the basestations are not at the extremes of the detectable signal strength spectrum. At these locations, noise can result in both positive or negative error, which allows for a greater number of possible signal strength values. Even if the recallable probability of each reading is less, the presence of more unique readings at a particular location can result in a darker color on a recall map.

The performance of the ACT-R algorithm differed from that of No Forgetting. Selected estimation error and recallable reading maps from applying the ACT-R forgetting algorithm to Path 2 are presented in Figure VIII.3. Figure VIII.1 demonstrates that during the second half of path 2, the ACT-R forgetting method experienced a large increase in estimation error for approximately 550 time units. This phenomena occurred because the ACT-R forgetting method began to filter readings located on the left side of the environment before the robot was able to complete the path. The ACT-R forgetting algorithm started to filter the earliest readings by the 648th time step and the robot had completed the first pass through the environment (Figure VIII.3a). The effects of filtering are visible for the sub-path from the center of the environment to the lower left corner of the environment and in the corner itself. A slight increase in estimation error can be seen in the lower left corner of Figure VIII.3i. As the robot continued the pass back to the left side of the environ-

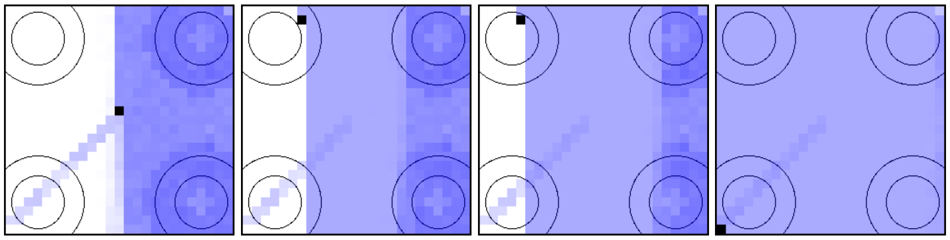


(a) Recall
648 time steps

(b) Recall
673 time steps

(c) Recall
799 time steps

(d) Recall
899 time steps

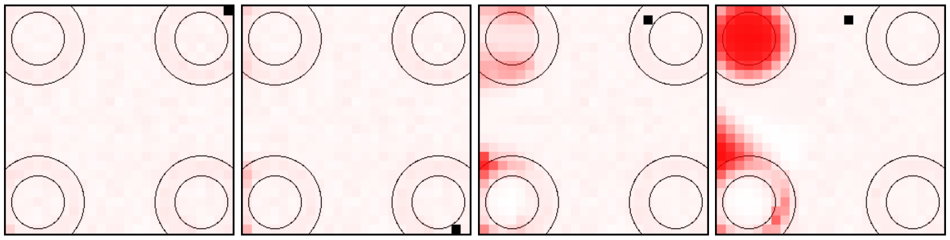


(e) Recall
959 time steps

(f) Recall
1099 time steps

(g) Recall
1149 time steps

(h) Recall
1272 time steps

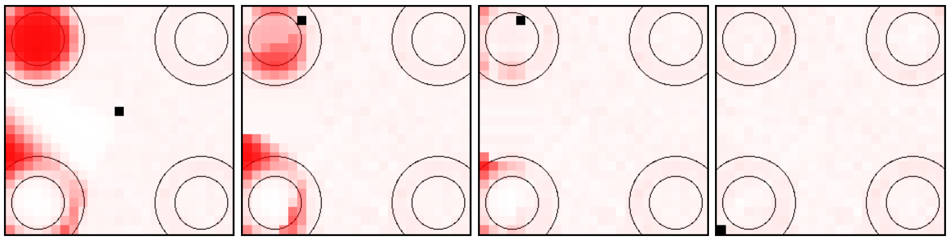


(i) Error
648 time steps

(j) Error
673 time steps

(k) Error
799 time steps

(l) Error
899 time steps



(m) Error
959 time steps

(n) Error
1099 time steps

(o) Error
1149 time steps

(p) Error
1272 time steps

Figure VIII.3: Path 2 - 1 cycle 4 basestations with ACT-R error and recall maps

ment, the effects of filtering increase and additional readings from the left side of the environment become unrecallable (Figures VIII.3b - VIII.3e). During this period, the estimation error present on the left side of the environment continued to increase (Figure VIII.3j - VIII.3m). After collecting more readings, the robot traveled past the transition point between where readings were recallable and where they were unrecallable (Figure VIII.3f). While Figures VIII.3n and VIII.3o do not show increasing error on the right side of the environment, Figures VIII.3f and VIII.3g show ACT-R continuing to filter greater numbers of old readings. At the end of the path, the error map (Figure VIII.3p) visually appears the same as the map generated by No Forgetting (Figure VIII.2p), but the recallable reading map shows significant differences. The map generated by No Forgetting (Figure VIII.2h) reveals large deviations in the number of recallable readings per location in the environment, while the recall map generated by ACT-R (Figure VIII.3h) suggests only one reading was available (except for the sub-path from the center of the environment to the lower left corner and the right most edge of the environment).

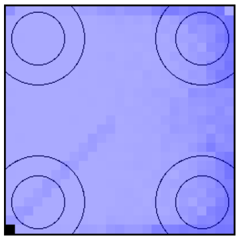
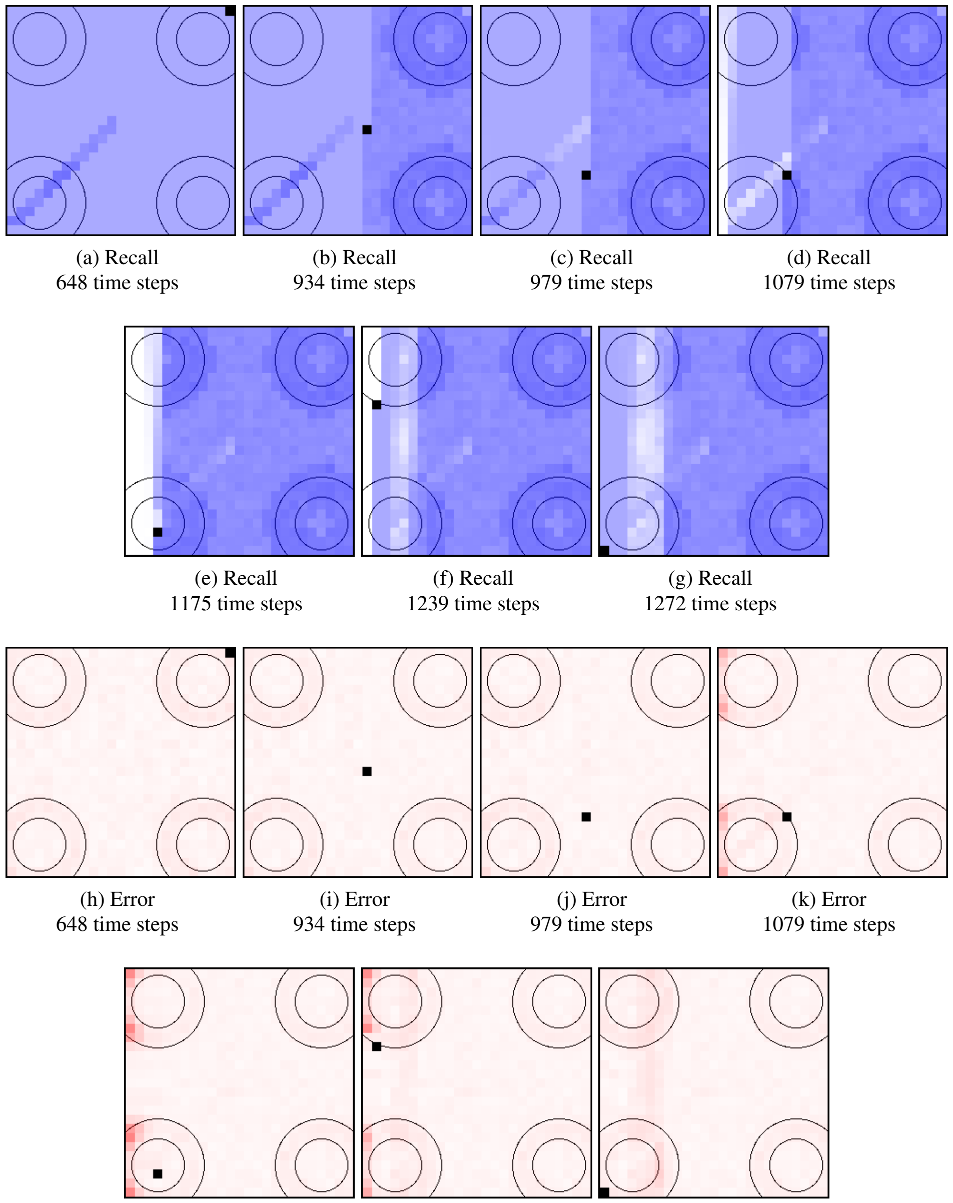


Figure VIII.4: End of path recall map for ActSimple and path 2 - 1 cycle 4 basestations

Similarity-base interference affected the number of recallable readings generated by the ActSimple algorithm. Figure VIII.4 presents the recallable reading map for ActSimple at the end of path 2. The top, bottom, and right edges of the environment exhibited larger numbers of recallable readings, unlike the left side of the environment. ActSimple's parameterization places a large emphasis on spatial similarity. Readings located on the edges of the environment have fewer nearby locations and consequently experienced reduced interference. Increased numbers of recallable readings are not available on the left edge of the environment due to the effects of time.



(a) Recall
648 time steps

(b) Recall
934 time steps

(c) Recall
979 time steps

(d) Recall
1079 time steps

(e) Recall
1175 time steps

(f) Recall
1239 time steps

(g) Recall
1272 time steps

(h) Error
648 time steps

(i) Error
934 time steps

(j) Error
979 time steps

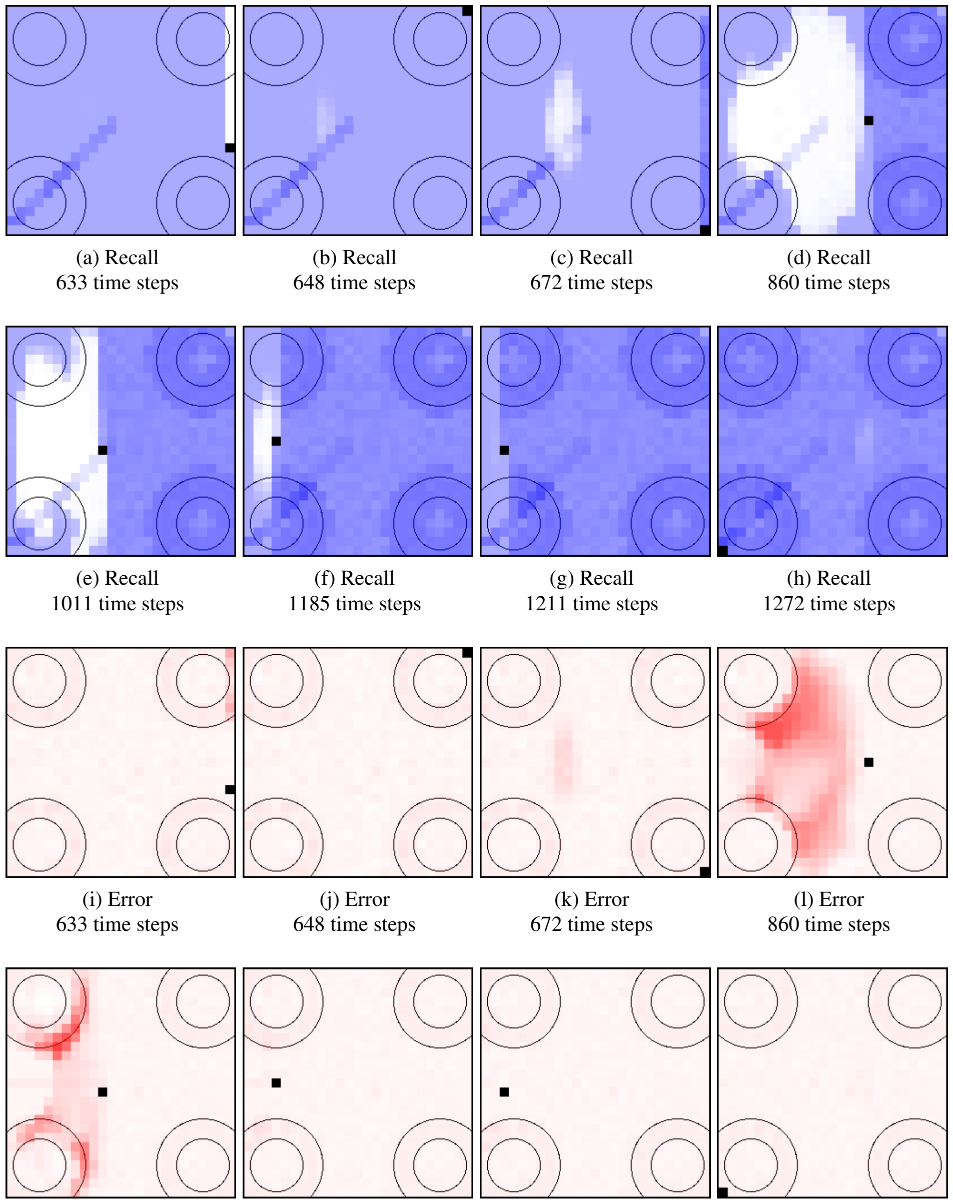
(k) Error
1079 time steps

(l) Error
1175 time steps

(m) Error
1239 time steps

(n) Error
1272 time steps

Figure VIII.5: Path 2 - 1 cycle 4 basestations with Queue Static error and recall maps


(a) Recall
633 time steps

(b) Recall
648 time steps

(c) Recall
672 time steps

(d) Recall
860 time steps

(e) Recall
1011 time steps

(f) Recall
1185 time steps

(g) Recall
1211 time steps

(h) Recall
1272 time steps

(i) Error
633 time steps

(j) Error
648 time steps

(k) Error
672 time steps

(l) Error
860 time steps

(m) Error
1011 time steps

(n) Error
1185 time steps

(o) Error
1211 time steps

(p) Error
1272 time steps

Figure VIII.6: Path 2 - 1 cycle 4 basestations with SIMPLE Update error and recall maps

The Queue Static algorithm's limited queue size had an effect on the forgetting algorithm's performance along Path 2 (Figure VIII.5). Since the queue size for Queue Static was 776, estimation error and recall were the same as No Forgetting for the collection of the first 776 unique readings (Figures VIII.5h, VIII.5i, VIII.5a, and VIII.5b). Around the 979th time step, the queue became full and older readings began to be forgotten. Figure VIII.5c shows the initial readings from the center of the environment starting to become unrecallable. However, the associated error map (Figure VIII.5j) did not reveal the effects of forgetting. The limited queue size's effects started to prevent readings from the left side of the environment from being recallable by the 1079th time step (Figure VIII.5d). The effects on estimation error began to appear at this time, increasing error in the affected left side of the environment (Figure VIII.5k). The primary region experiencing the effects of the limited queue size began moving to the right of the environment as the robot continued to follow the path. A band of limited reading recallability can be seen in Figures VIII.5e - VIII.5g. The effects on estimation error can be seen in the corresponding error maps (Figures VIII.5l - VIII.5n).

The SIMPLE Update algorithm experienced a substantial increase in estimation error similar to ACT-R (Figure VIII.1). Selected recall and error maps for the SIMPLE Update algorithm being applied to path 2 are presented in Figure VIII.6. The SIMPLE Update algorithm's performance was visually identical to No Forgetting for the first 633 time steps (Figures VIII.6a and VIII.6i). The performance of SIMPLE Update began to diverge and by the 648th time step, the forgetting algorithm began filtering readings in the center of the environment (Figure VIII.6b), but an increase in estimation error was not visible in the associated error map (Figure VIII.6j). Additional readings began to be filtered (Figure VIII.6c) and visible increases in estimation error started to occur as the robot began the second half of the path (Figure VIII.6k). The filtering of readings continued in the center of the environment (Figure VIII.6d) and estimation error also continued to increase (Figure VIII.6l). Continuing along the path, the robot progressed to the left side of the environment, traveling through the area experiencing the filtering of readings (Figure VIII.6e), reducing estimation error (Figure VIII.6m). The robot then collected additional readings, which removed the substantial filtering of readings that occurred in the center of the environment (Figures VIII.6f and VIII.6g). These additional readings removed the visible increase in estimation error and after completing a total of 1211 time steps, the elevated estimation error was no longer visible (Figure VIII.6o). A second increase in the filtering of readings started to appear at the end of the path (Figure VIII.6h). However, a second visible increase in estimation error did not occur (Figure VIII.6p).

VIII.1.2 Performance of Forgetting when Applied to Path 4

Path 4 is similar to path 2 in that it involves one cycle through the environment and the four basestation configuration, but differs in the number of locations visited in the environment's upper left corner. Figure VIII.7

presents path 4's trajectory (repeated from Figure IV.11d). When the robot travels into the upper left section, the collection of readings is not exhaustive and readings are only collected for the locations visited by the robot. This section describes the performance differences with ACT-R and ActSimple that result from the change in path trajectory from Path 2 to Path 4.

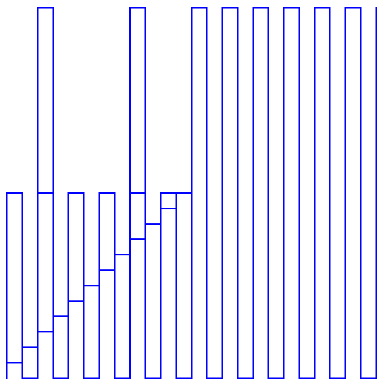


Figure VIII.7: Trajectory of path 4 (Repeated from Figure IV.11d

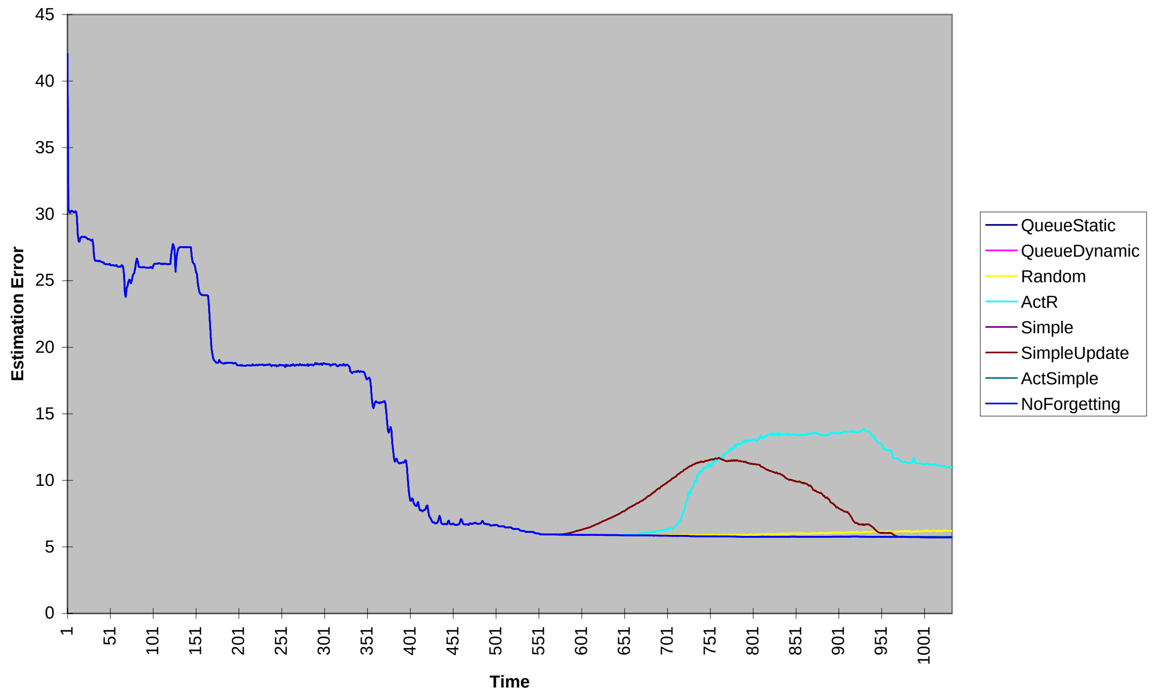
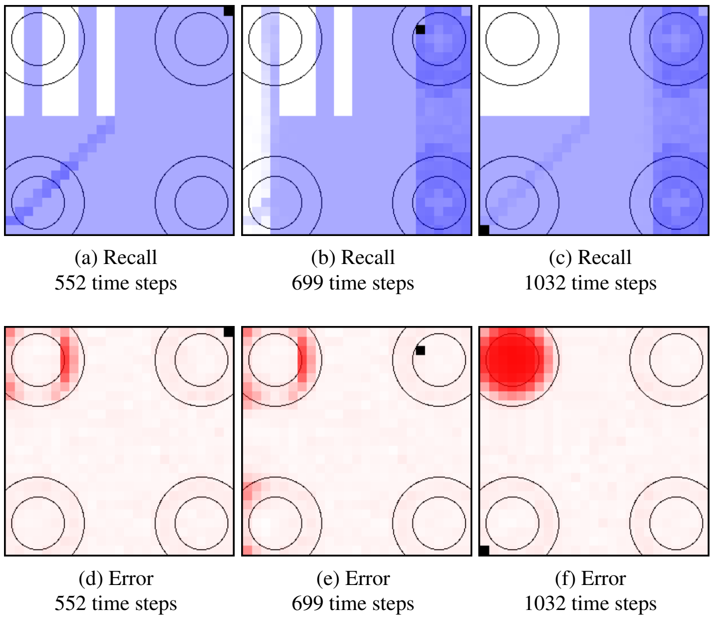


Figure VIII.8: Along path results for path 4 - 1 cycle 4 basestations

The ACT-R algorithm's end of path estimation error with Path 4 (time 1033 in Figure VIII.8) increased substantially from path 2 (time 1272 in Figure VIII.1). ACT-R generated an end of path estimation error


(a) Recall
552 time steps

(b) Recall
699 time steps

(c) Recall
1032 time steps

(d) Error
552 time steps

(e) Error
699 time steps

(f) Error
1032 time steps

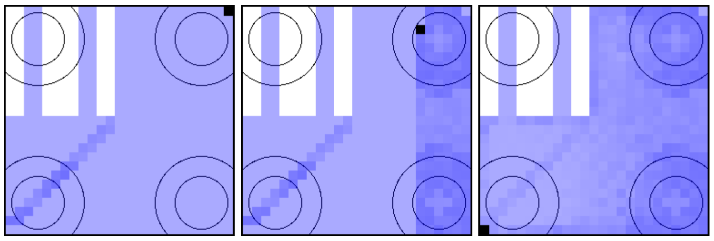
Figure VIII.9: Path 4 - 1 cycle 4 basestations with ACT-R error and recall maps

of 4.703 for path 2, but the Path 4 results have an end of path error of 10.950. Figure VIII.9 presents a subset of the error and recall maps for ACT-R when applied to path 4. Similar to the filtering of the readings that occurred along path 2, ACT-R filtered the readings collected from the left side of the environment, including the readings from the upper left section (Figure VIII.9b). All of the readings from the upper left were unrecallable (Figure VIII.9c) and significant estimation error was generated (Figure VIII.9f) by the time the robot returned to the lower left corner of the environment.

ActSimple's performance differed from ACT-R's along path 4. Figure VIII.10 presents the recallable reading and error maps for ActSimple along path 4. After 699 time steps, ActSimple did not visually begin to filter the readings gathered in the left half of the environment (Figure VIII.10b). However, bands of large estimation error were still present (Figure VIII.10e), similar to ACT-R. When the robot completed the path, the readings located in the upper left section were still not being filtered (Figure VIII.10c). However, some readings in the bottom left portion do appear to have been filtered, since this area is a lighter more consistent blue than the right portion. Unlike ACT-R, ActSimple generated considerably less error in the upper left portion of the environment (Figure VIII.10f).

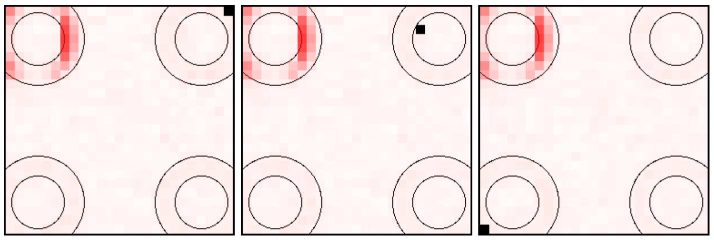
VIII.1.3 One Cycle, Eight Basestation Paths

Estimation error along the three paths involving one cycle and eight basestations was similar to the performance observed along the one cycle, four basestation paths except for the effects of the first acquired


(a) Recall
552 time steps

(b) Recall
699 time steps

(c) Recall
1032 time steps


(d) Error
552 time steps

(e) Error
699 time steps

(f) Error
1032 time steps

Figure VIII.10: Path 4 - 1 cycle 4 basestations with ActSimple error and recall maps

reading and the initial rate of error reduction. The eight basestation paths maintained a larger percentage of the environment where strong signal strength was available. As a result, the estimation error generated before any readings were collected was 33.999. Unlike the four basestation paths described in Chapters VIII.1.1 and VIII.1.2, which experienced an estimation error drop from 42.070 to 30.041, the error increased to 57.562 after the first reading was collected. This phenomena is present in the graph of estimation error for path 8 (Figure VIII.11). Paths 6 and 7 generated similar estimation error as path 8 and are presented in Figures A.2 and A.3.

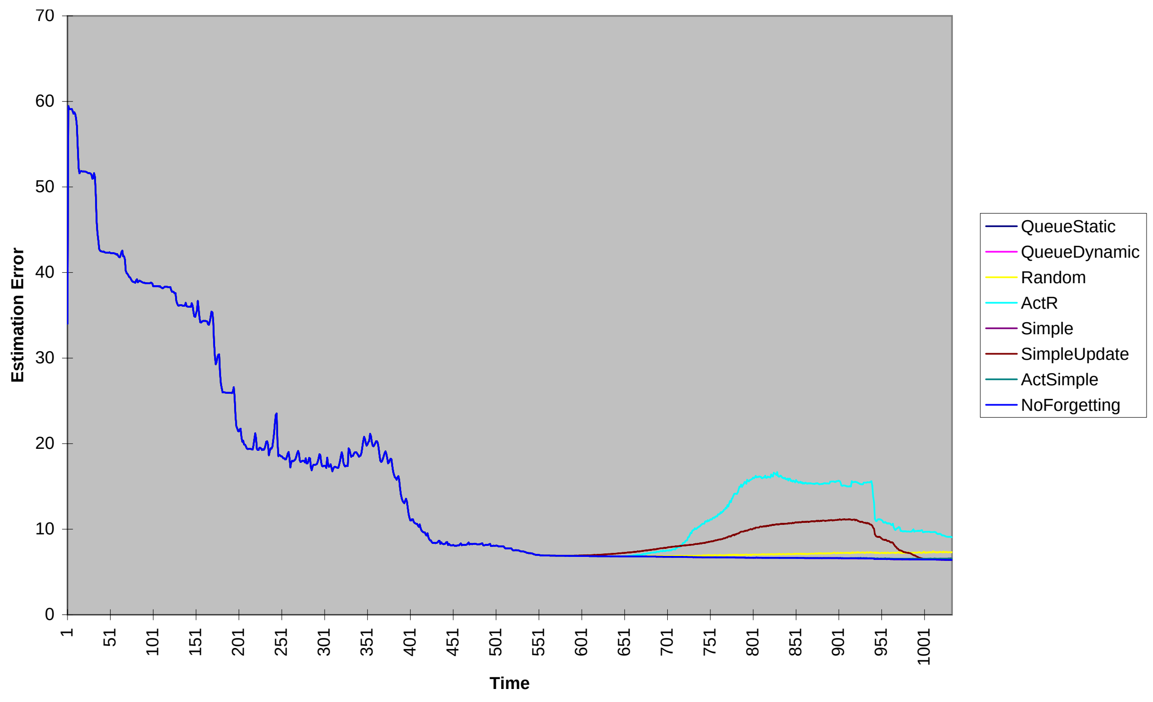


Figure VIII.11: Along path results for path 8 - 1 cycle 8 basestations

VIII.1.4 Static, Three Cycle Paths

The performance of the forgetting algorithms and No Forgetting on the beginning portions of the three cycle, single basestation configuration (either four or eight) paths (paths 10-12 and 14-16) were similar to their complementary one cycle paths (paths 2-4 and 6-8). The performance of each forgetting algorithm and No Forgetting started to diverge as the second cycle in each path began. Figure VIII.12 presents the estimation error along path 10, which is representative of the estimation error values diverging.

The SIMPLE Update algorithm generated an oscillating estimation error along path 10. Selected recall maps for SIMPLE Update applied to path 10 are presented in Figure VIII.13. The complimentary error maps are presented in Figure VIII.14. Path 10 includes three cycles through the environment, all of which are identical except for an initial sub-path from the center of the environment to the lower left corner. The first cycle in path 10 is identical to the entirety of path 2 and the performance of SIMPLE Update along path 10 initially was indistinguishable to the results of applying SIMPLE Update to path 2 (Figures VIII.1 and VIII.6). A description of SIMPLE Update applied to path 2 is presented in Chapter VIII.1.1. Figures VIII.13a - VIII.13b and Figures VIII.14a - VIII.14b show the recallable readings and the estimation error generated by the SIMPLE Update algorithm for the final portion of the first cycle through the environment. At the conclusion of the first cycle (after 1272 time steps), a second instance of significant filtering started to appear in the

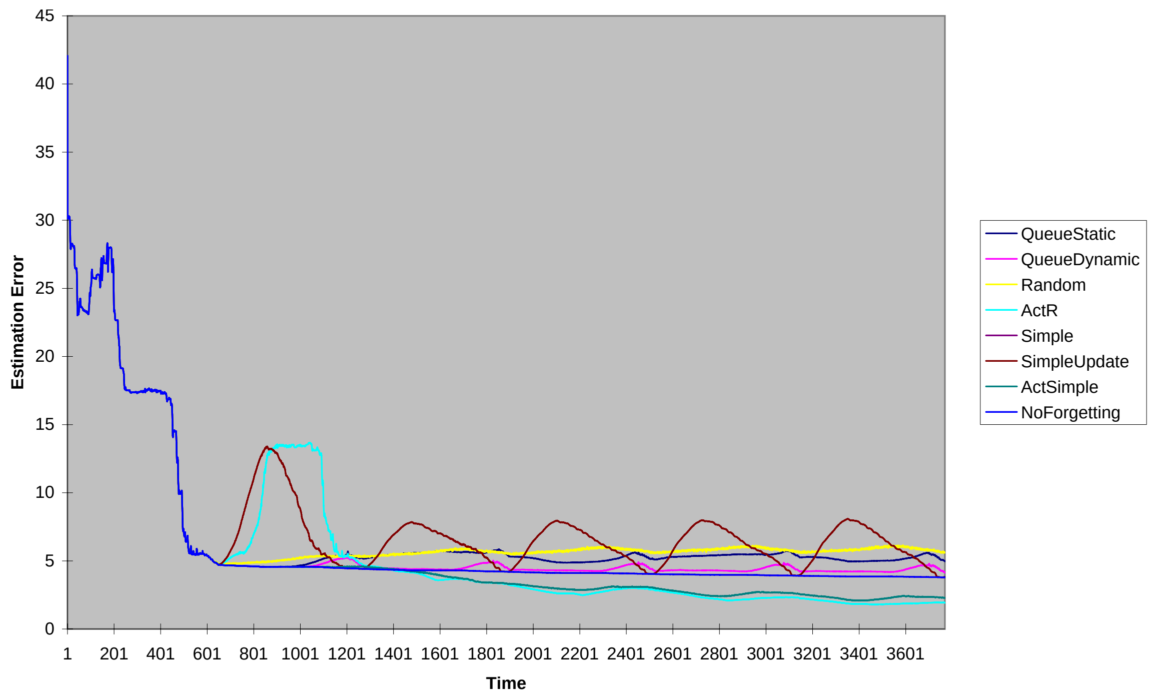
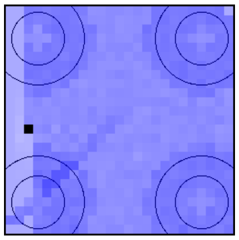
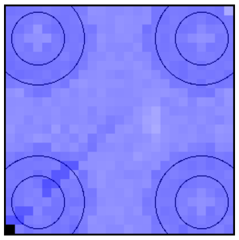
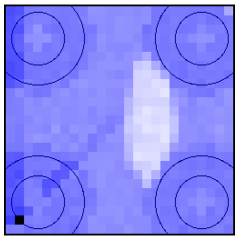


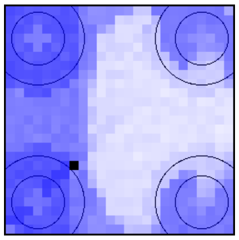
Figure VIII.12: Along path results for path 10 - 3 cycles 4 basestations

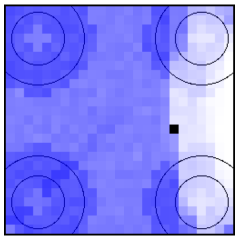
middle of the environment (Figure VIII.13b). However, an increase in estimation error was not visible (Figure VIII.14b). The robot began the second cycle. After a total of 1320 time steps, the region experiencing substantial filtering had expanded (Figure VIII.13c) and an increase in estimation error was observed (Figure VIII.14c). The area where readings were being filtered continued to expand, while the robot completed the 1464th time step. Estimation error increased in a similar fashion (Figure VIII.14c). After the 1733th time step, the robot had ventured into the filtering area, reducing the area's size (Figure VIII.13e). While the size of the filtered reading area was shrinking, new readings were being filtered at the right edge of the environment. Estimation error along the right side of the environment similarly increased (Figure VIII.14e). The robot finished the first half of the second cycle by completing the 1896th time step while traveling to the upper right corner. The newly collected readings on the right side of the environment reduced the effects of filtering on that half of the environment (Figure VIII.13f) and elevated estimation error was also removed (Figure VIII.14f). During the robot's return trip to the left side of the environment, another area of extensive filtering appeared (Figure VIII.13g) and estimation error also increased (Figure VIII.14g). After the 2520th time step, the filtering and increase in estimation error had again subsided (Figures VIII.13h and VIII.14h). The third cycle through the environment was similar to the second cycle (Figures VIII.13i - VIII.13l and Figures VIII.14i - VIII.14l).

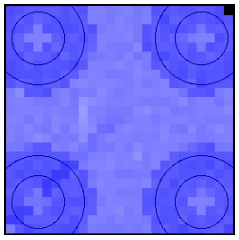

(a) Recall
1211 time steps

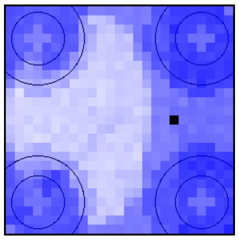

(b) Recall
1272 time steps

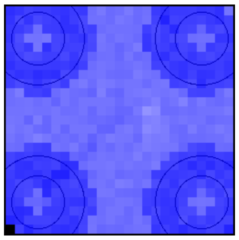

(c) Recall
1320 time steps

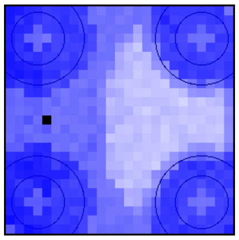

(d) Recall
1464 time steps

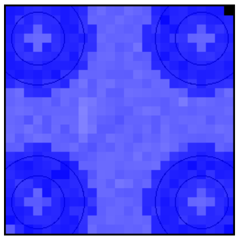

(e) Recall
1733 time steps

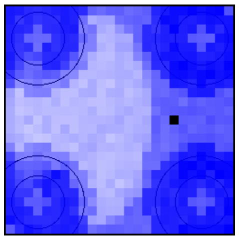

(f) Recall
1896 time steps


(g) Recall
2058 time steps


(h) Recall
2520 time steps


(i) Recall
2632 time steps


(j) Recall
3144 time steps


(k) Recall
3306 time steps

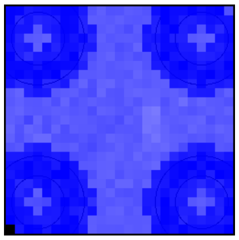
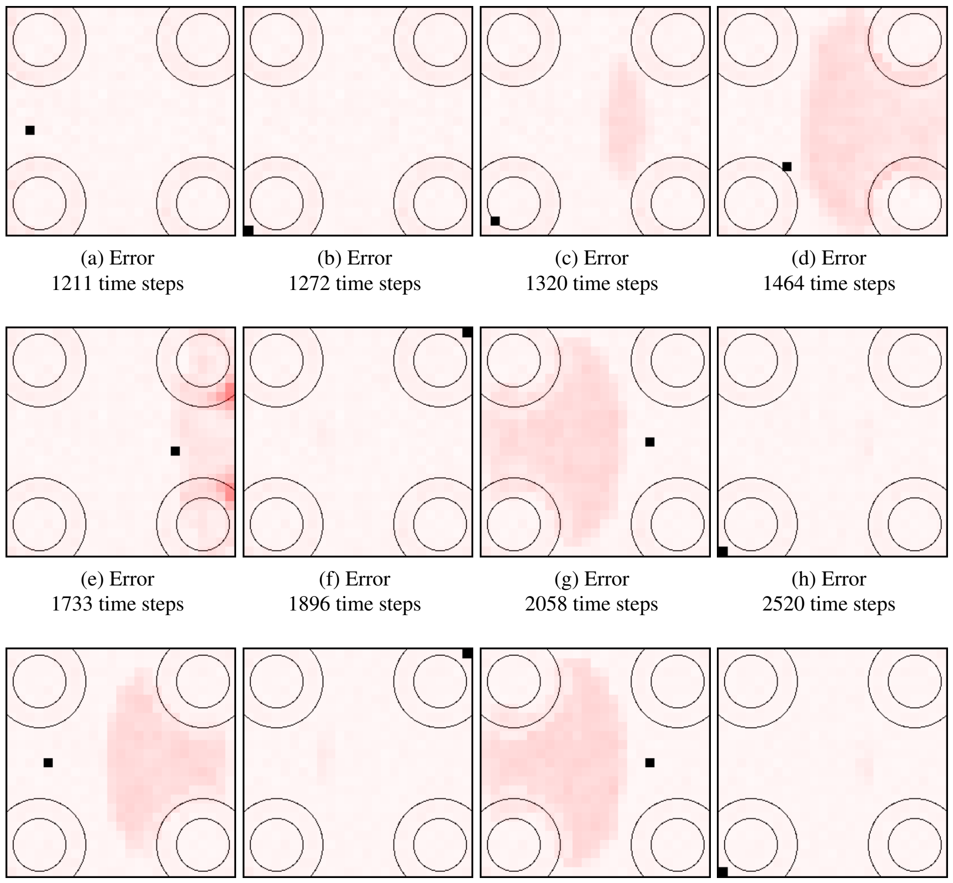

(l) Recall
3768 time steps

Figure VIII.13: Path 10 - 3 cycles 4 basestations with SIMPLE Update recall maps



(a) Error
1211 time steps

(b) Error
1272 time steps

(c) Error
1320 time steps

(d) Error
1464 time steps

(e) Error
1733 time steps

(f) Error
1896 time steps

(g) Error
2058 time steps

(h) Error
2520 time steps

(i) Error
2632 time steps

(j) Error
3144 time steps

(k) Error
3306 time steps

(l) Error
3768 time steps

Figure VIII.14: Path 10 - 3 cycles 4 basestations with SIMPLE Update error maps

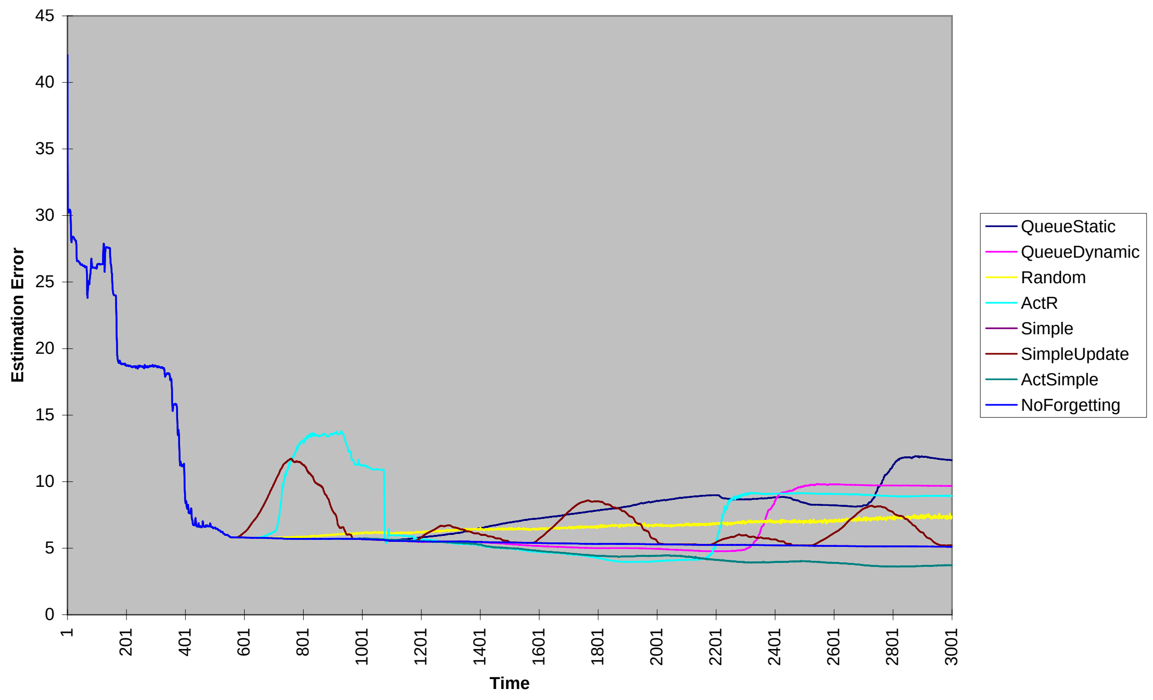


Figure VIII.15: Along path results for path 12 - 3 cycles 4 basestations

The differences in estimation error were greater along path 12 (Figure VIII.15). Near the end of the path, Queue Static, Queue Dynamic, and ACT-R each experienced a "step-like" increase in error. Path 12's trajectory (Figure VIII.16, repeated from Figure IV.11l) is very similar to path 4's trajectory (Figure VIII.7, repeated from Chapter IV.4). The first two cycles in the path 12 trajectory each reproduce path 4's trajectory, except the second cycle does not include the sub-path from the center of the environment to the lower left corner. The third cycle is similar to the second cycle, except the robot does not travel into the upper left quadrant of the environment. The second and third cycles only visit locations previously visited by the first cycle, resulting in the graphical representation of path 12's trajectory (Figure VIII.16) being identical to the graphical representation for path 4's trajectory (Figure VIII.7).

The large estimation error deviations and the "step-like" trends in the last third of path 12 result from the interaction of path 12's trajectory with the upper left corner of the environment (similar to the interaction experienced by path 4 and that quadrant of the environment). Figure VIII.17 presents selected recall and error maps for ACT-R when applied to path 12. The number of recallable readings in each location along path 12 (Figures VIII.17a - VIII.17c) were identical to the results from path 4 (Figures VIII.9a - VIII.9c) for the first 1032 time steps. The error maps for path 12 (Figures VIII.17i - VIII.17k) were also identical to the error maps from path 4 (Figures VIII.9d - VIII.9f) during this period of time. The robot then began the second

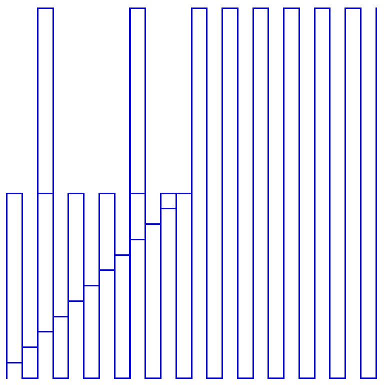


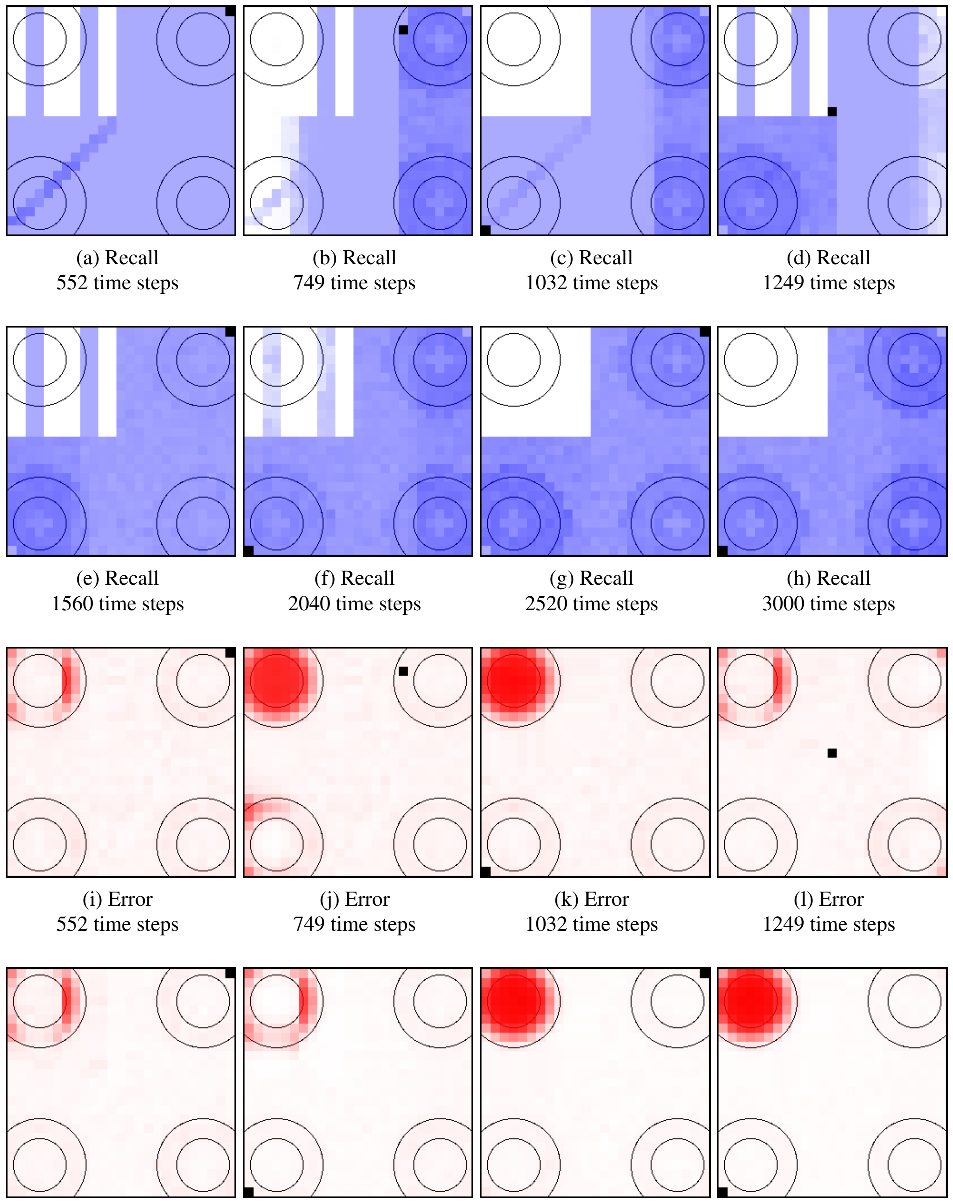
Figure VIII.16: Trajectory of path 12 (Repeated from Figure IV.11l)

cycle, returning to the upper left quadrant (Figure VIII.17d). While readings were again recallable in the upper left, readings at the right edge of the environment were being filtered. The newly recallable readings in the upper left reduced the estimation in that area, but an increase in estimation error was present on the right side of the environment due to readings being filtered (Figure VIII.17l). The robot continued along the path and after 1560 time steps, the error map (Figure VIII.17m) resembled the error map from mid-point of the first cycle (Figure VIII.17i). The recall map (Figure VIII.17e) reveals readings were available throughout the entire trajectory. The robot returned to the lower left corner and the readings in the upper left quadrant again began to become unrecallable (Figure VIII.17f), resulting in slightly increased error (Figure VIII.17n). During the third cycle, the robot did not return to the upper left quadrant and the readings from that location were filtered (Figures VIII.17g and VIII.17h). Substantial error returned to the upper left portion of the environment (Figure VIII.17o and VIII.17p).

The Queue Static and Queue Dynamic algorithms experienced similar "step-like" increases in estimation error due to similar interactions with the upper left quadrant of the environment. While Random forgetting also experienced increasing error during the second half of this path, Random forgetting did not generate a large jump in estimation error like the other queue-based forgetting methods or ACT-R. The results for paths 11, 14, 15, and 16 are located in Appendix A.

VIII.1.5 Dynamic, Non-Random Paths

When the forgetting algorithms and No Forgetting were applied to the non-random paths containing a basestation configuration change, the resultant estimation error during the first half of each path was similar to the performance seen on the static paths. Once the basestation configuration change occurred, the estimation error generated by the forgetting algorithms and No Forgetting increased sharply. The change in estimation



(a) Recall
552 time steps

(b) Recall
749 time steps

(c) Recall
1032 time steps

(d) Recall
1249 time steps

(e) Recall
1560 time steps

(f) Recall
2040 time steps

(g) Recall
2520 time steps

(h) Recall
3000 time steps

(i) Error
552 time steps

(j) Error
749 time steps

(k) Error
1032 time steps

(l) Error
1249 time steps

(m) Error
1560 time steps

(n) Error
2040 time steps

(o) Error
2520 time steps

(p) Error
3000 time steps

Figure VIII.17: Path 12 - 3 cycles 4 basestations with ACT-R error and recall maps

error is presented in Figure VIII.18 for path 5, Figure VIII.19 for path 9, Figure VIII.20 for path 13, and Figure VIII.21 for path 1. The basestation configuration change occurred after 648 time steps for paths 1 and 5 and after 1896 time steps for paths 9 and 13. The effects of the basestation configuration change were different for each path, and estimation error from the individual forgetting algorithms and No Forgetting diverged. While the forgetting algorithms generated worse estimation error than No Forgetting for at least a portion of each path after the basestation configuration change, all forgetting algorithms, except SIMPLE produced less estimation error than No Forgetting at the end of each path (Figures VIII.21 - VIII.20). The Queue Static, Queue Dynamic, and ACT-R algorithms generated substantially greater error than No Forgetting along path 1 after the basestation configuration change. The Queue Static and Queue Dynamic algorithms' periods of increased error lasted for approximately 110 time steps, while the ACT-R algorithm's increase lasted for roughly 185 time steps.

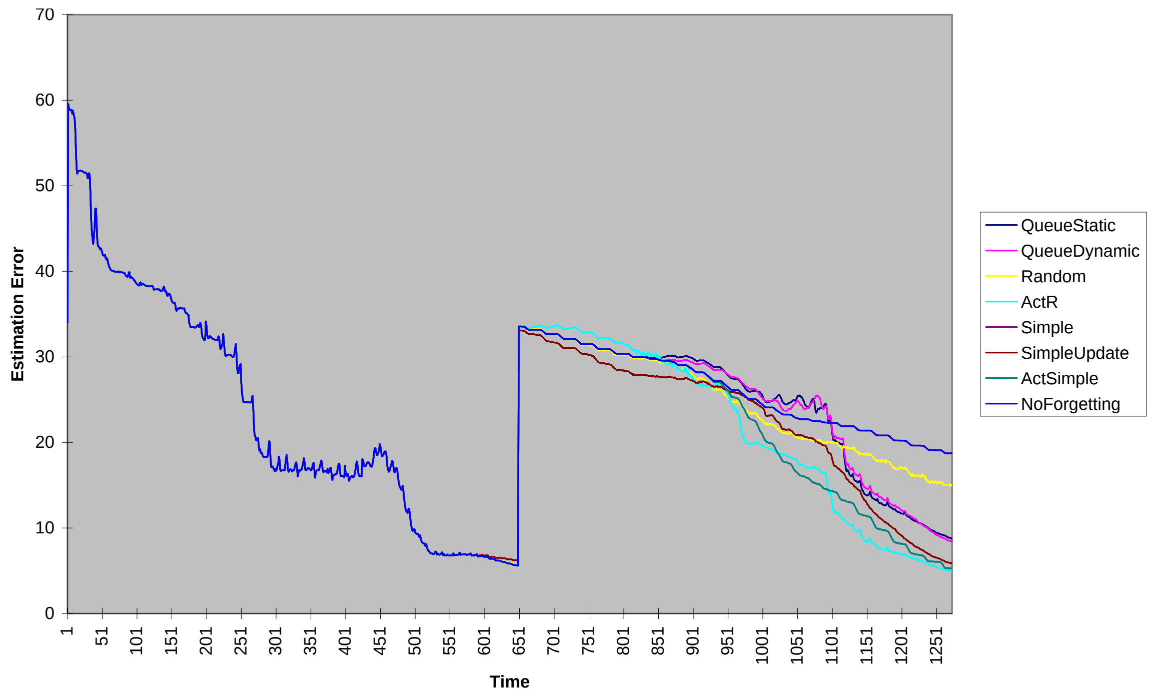


Figure VIII.18: Along path results for path 5 - 1 cycle 8 - 4 basestations

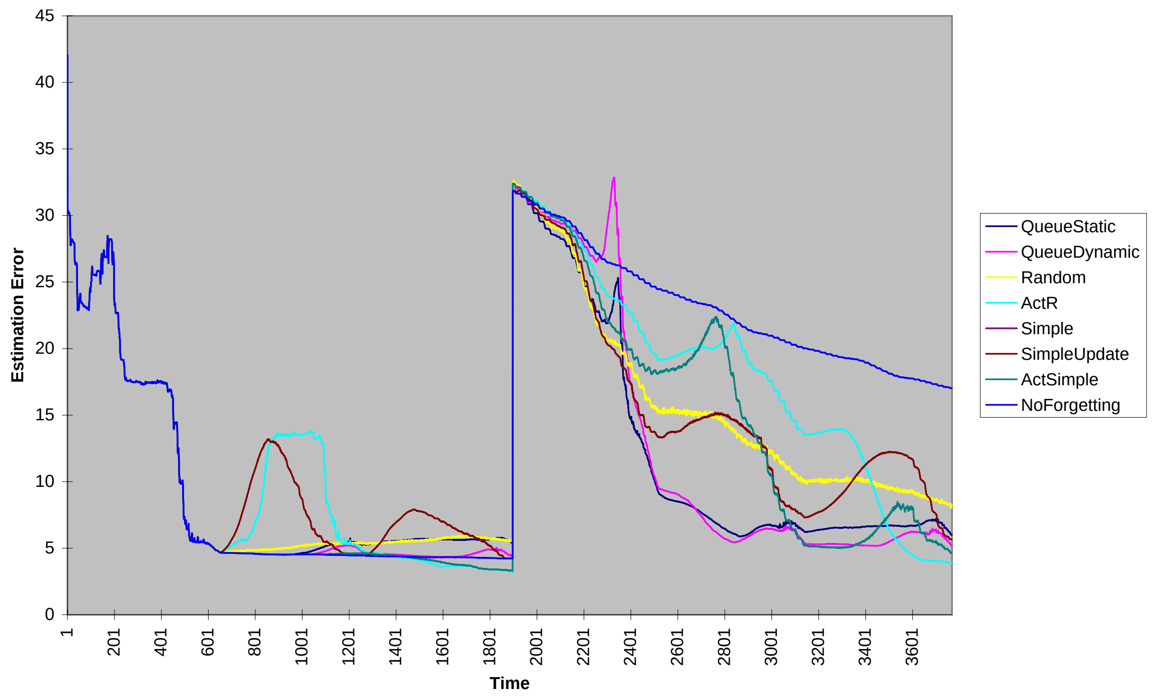


Figure VIII.19: Along path results for path 9 - 3 cycles 4 - 8 basestations

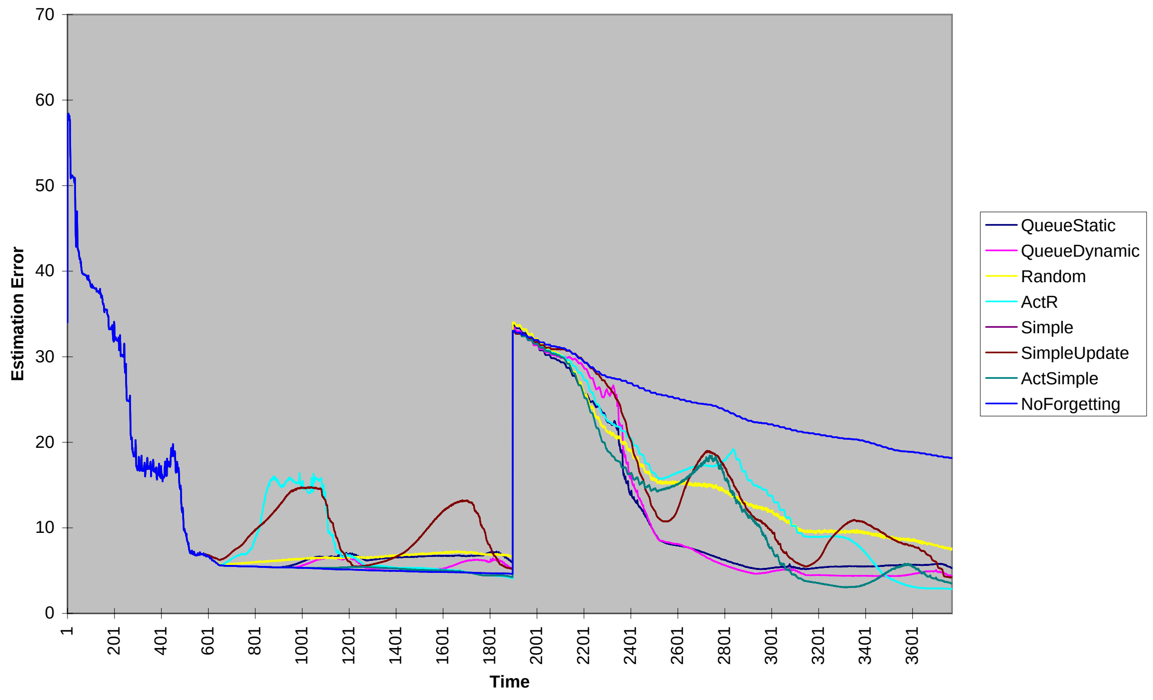


Figure VIII.20: Along path results for path 13 - 3 cycles 8 - 4 basestations

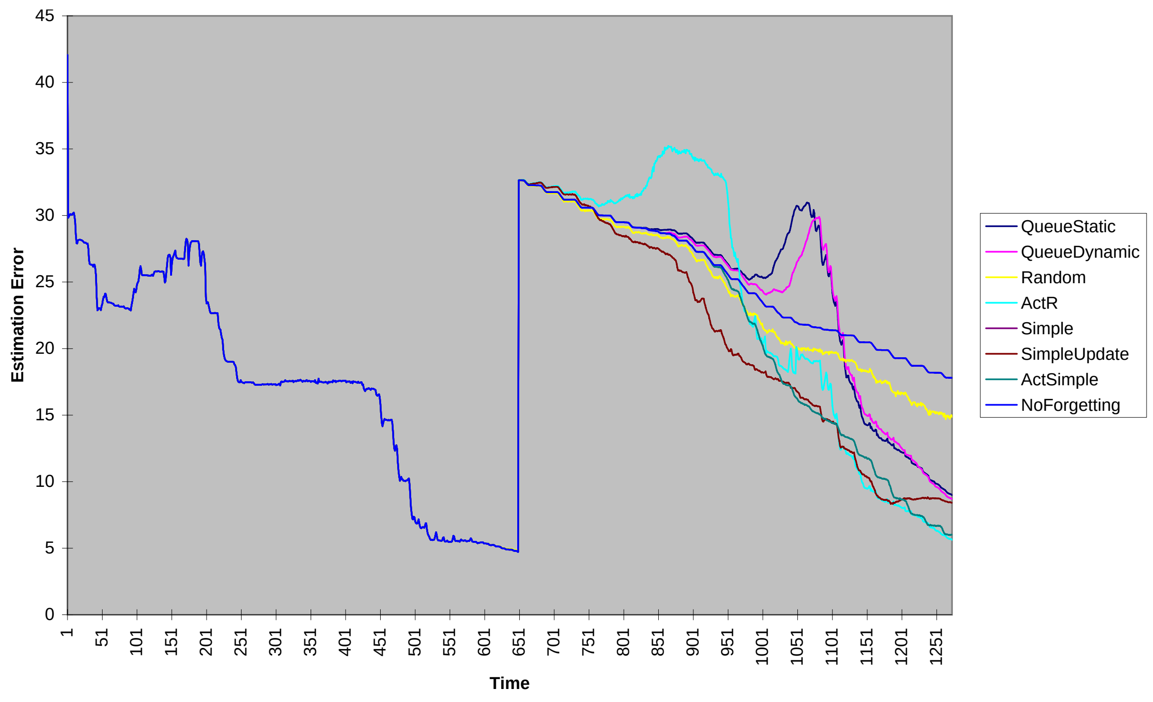
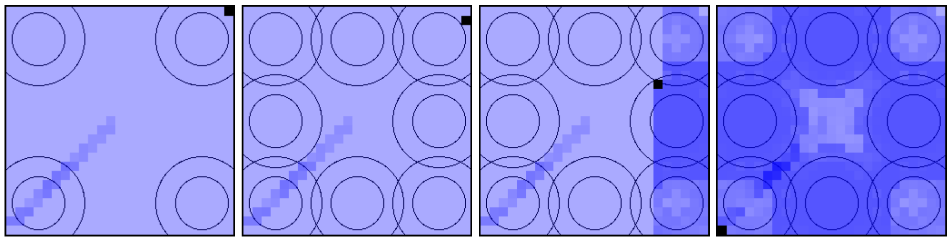


Figure VIII.21: Along path results for path 1 - 1 cycle 4 - 8 basestations

Basestation configuration changes can have a significant impact on the amount of estimation error present at each location within the environment. The estimation error and recallable reading maps for No Forgetting when applied to path 1 are presented in Figure VIII.22. No Forgetting performed the same as with path 2 (Figure VIII.2) for the first 648 time steps (Figures VIII.22a and VIII.22e), but the estimation error changed abruptly after the basestation configuration change and the 649th time step (Figure VIII.22f). The basestation configuration switched from the four basestation configuration to the eight basestation configuration and altered the error associated with each previously collected reading. New readings were collected (Figure VIII.22c) as the robot began the return trip to the left side, which aided in reducing the error in the locations near the new basestations (Figure VIII.22g). The error levels did not change near the old basestation location because the change in basestation configuration only added basestations, none were removed. At the end of the path, a new reading had been collected for nearly every location within the environment (Figure VIII.22d), helping to reduce error; however, No Forgetting was unable to remove the effects of the readings collected prior to the basestation configuration change (Figure VIII.22h).

VIII.1.6 Randomly Generated Paths

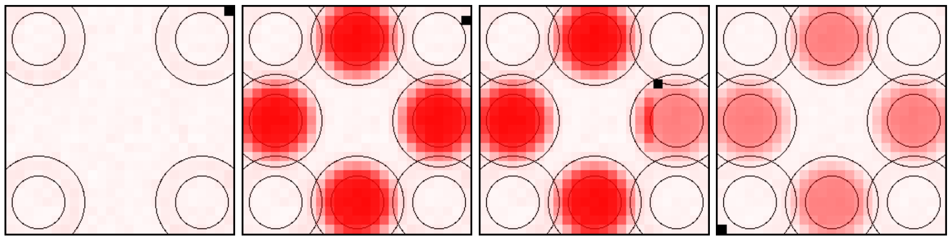
Four of the tested paths (paths 17 - 20) involved randomly generated trajectories. Two paths contained a basestation configuration change from either four to eight or eight to four basestations, while the remaining


(a) Recall
648 time steps

(b) Recall
649 time steps

(c) Recall
789 time steps

(d) Recall
1272 time steps


(e) Error
648 time steps

(f) Error
649 time steps

(g) Error
789 time steps

(h) Error
1272 time steps

Figure VIII.22: Path 1 - 1 cycle 4 - 8 basestations with No Forgetting error and recall maps

two paths only possessed a single basestation configuration. The estimation error performance along these paths resembled the performance observed on non-randomly generated paths (involving the same length and basestation configuration characteristics) except that the randomly generated paths tended to have sharper estimation error transitions. Figure VIII.23 shows the estimation error for path 17, a randomly generated path with a basestation configuration change after 1500 time steps and Figure VIII.24 presents performance along path 20, a path without a basestation configuration change. The performance along path 19 can be found in Figure A.8.

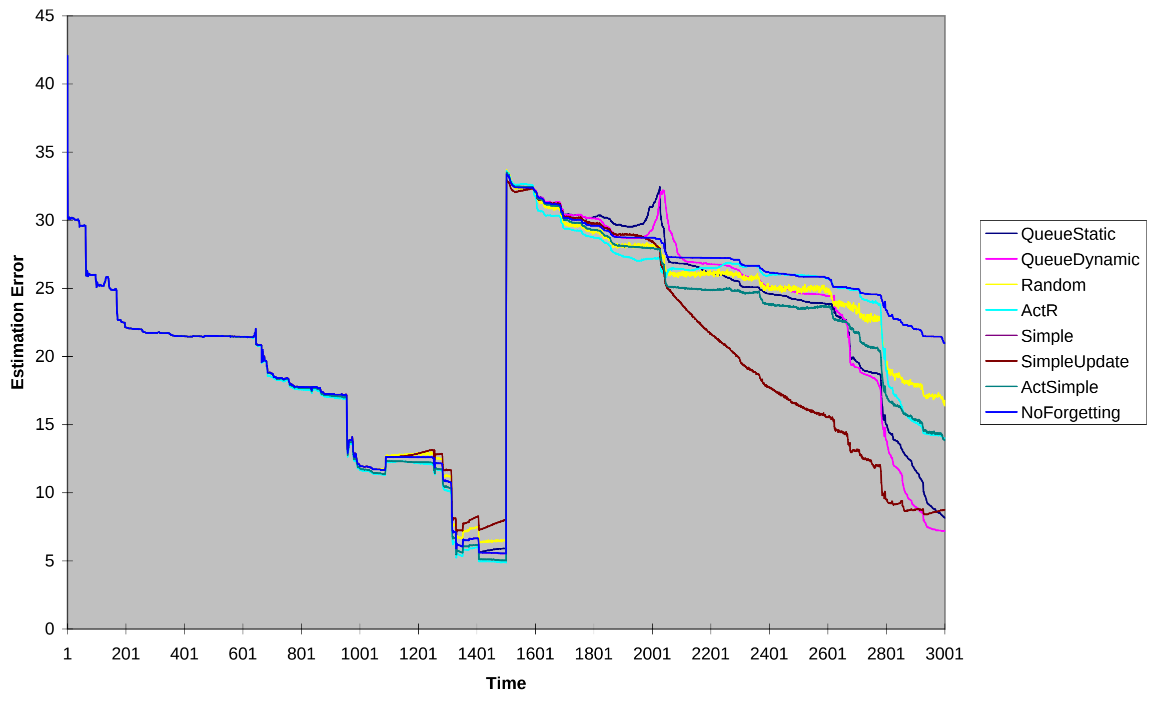


Figure VIII.23: Along path results for path 17 - random 4 - 8 basestations

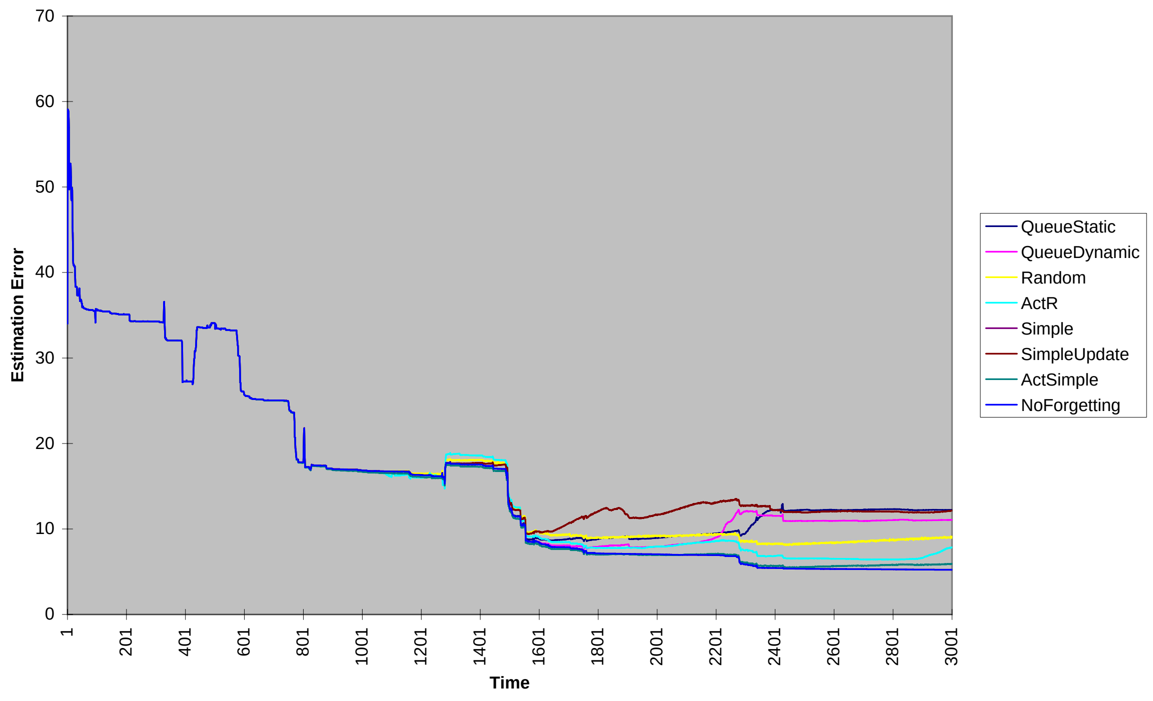


Figure VIII.24: Along path results for path 20 - random 8 basestations

The SIMPLE Update algorithm performed differently on path 18 (Figure VIII.25). Unlike the other tested forgetting algorithms or No Forgetting, SIMPLE Update experienced a large increase in estimation error from around time step 1500 to time step 2400. Figure VIII.26 presents selected recall and estimation error maps for SIMPLE Update when applied to path 18. After the first reading was collected (Figure VIII.26a), the SIMPLE Update algorithm generated the same amount of estimation error as every other forgetting algorithm (and No Forgetting) when applied to a path beginning in the 4 basestation configuration. After the first 65 time steps (Figure VIII.26b), the estimation error near the upper left basestation was reduced (Figure VIII.26j). After the 294th time step (Figure VIII.26c), the estimation error in the lower left corner was reduced (Figure VIII.26k). The majority of significant estimation error was removed (Figure VIII.26l) by the time of the 1050th time step (Figure VIII.26d). The robot continued collecting readings and by the 1552th time step, readings in the upper center of the environment started being filtered (Figure VIII.26e). Estimation error in this area of the environment started to increase (Figure VIII.26m). The number of readings being filtered continued to increase and by the 1970th time step, the majority of the center of the environment was being filtered (Figure VIII.26f). Estimation error similarly increased (Figure VIII.26n). The robot then traveled through the area where filtering was occurring and collected new readings (Figures VIII.26g and VIII.26h). These new readings resulted in the reduction of the increased estimation error (Figure VIII.26o and VIII.26p).

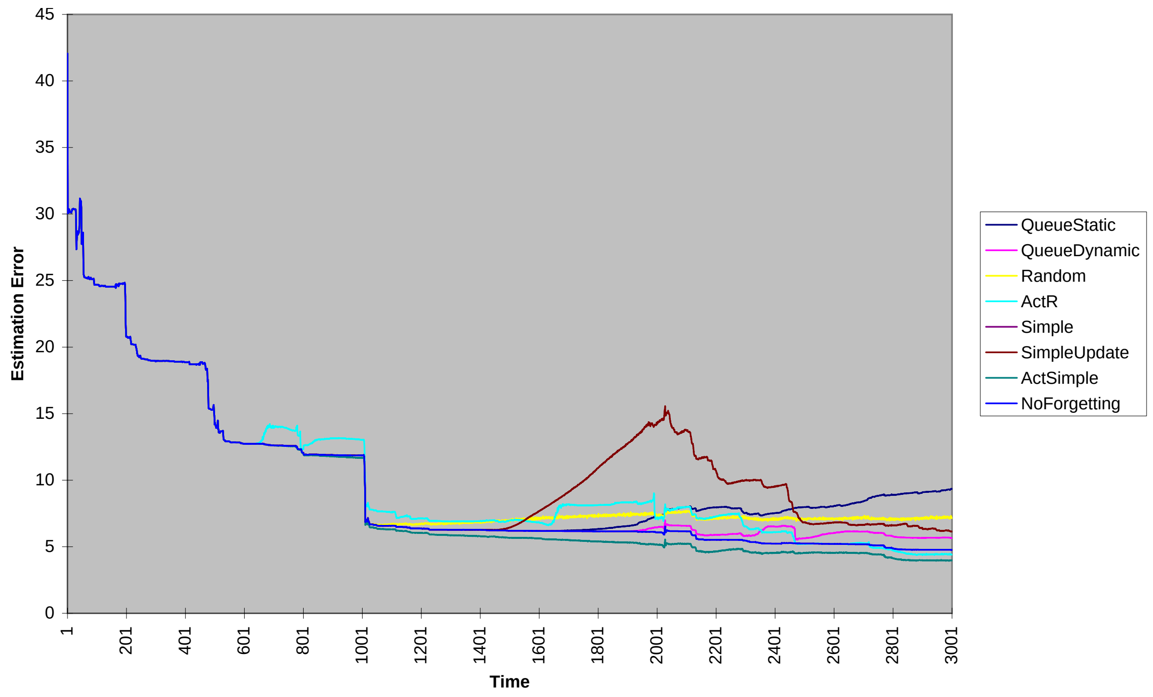
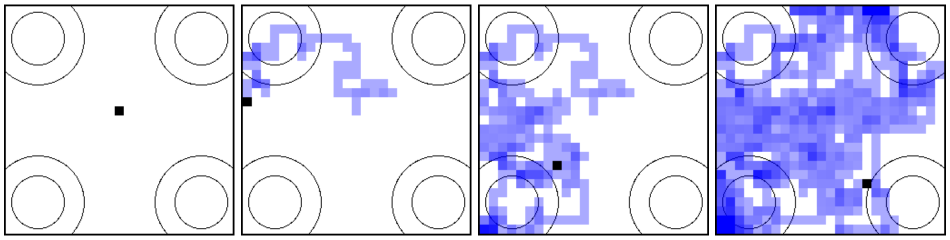
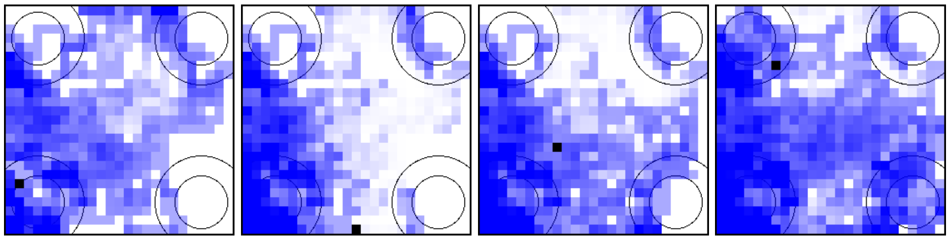


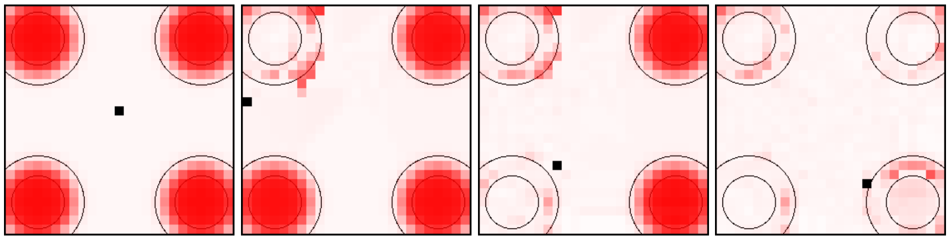
Figure VIII.25: Along path results for path 18 - random 4 basestations



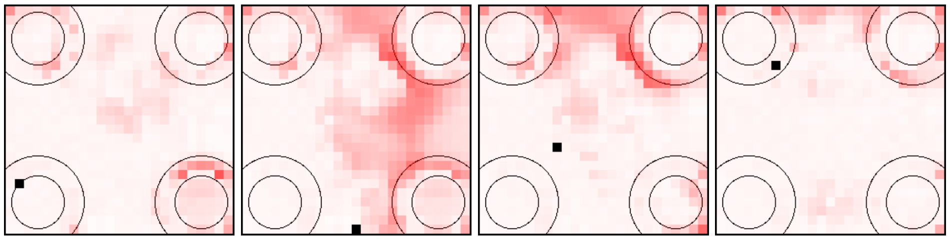
(a) Recall 1 time step | (b) Recall 65 time steps | (c) Recall 294 time steps | (d) Recall 1050 time steps



(e) Recall 1552 time steps | (f) Recall 1970 time steps | (g) Recall 2221 time steps | (h) Recall 3000 time steps



(i) Error 1 time step | (j) Error 65 time steps | (k) Error 294 time steps | (l) Error 1050 time steps



(m) Error 1552 time steps | (n) Error 1970 time steps | (o) Error 2221 time steps | (p) Error 3000 time steps

Figure VIII.26: Path 18 - random 4 basestations with SIMPLE Update recall and error maps

VIII.2 Along the Path Performance Metrics

The graphs from Chapter VIII.1 show varied relative performance along each tested path. In many domains and tasks, the precise time when a decision will need to be made will often not be known in advance. Ad-

Table VIII.1: Along path average estimation error

Path	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 1 - 1 Cycle 4 - 8 Basestations	21.09	20.83	20.70	20.47	20.08	21.09	18.80	19.25
Path 2 - 1 Cycle 4 Basestations	10.97	11.10	11.04	11.23	13.12	10.97	12.59	10.98
Path 3 - 1 Cycle 4 Basestations	11.62	11.63	11.62	11.74	13.37	11.62	12.75	11.63
Path 4 - 1 Cycle 4 Basestations	11.98	11.98	11.98	12.07	14.02	11.98	13.15	11.98
Path 5 - 1 Cycle 8 - 4 Basestations	24.65	23.87	23.88	23.93	22.41	24.65	22.85	22.56
Path 6 - 1 Cycle 8 Basestations	14.36	14.68	14.58	14.80	16.97	14.36	16.81	14.40
Path 7 - 1 Cycle 8 Basestations	15.00	15.04	15.01	15.21	17.00	15.00	15.94	15.02
Path 8 - 1 Cycle 8 Basestations	15.50	15.51	15.50	15.67	17.50	15.50	16.39	15.51
Path 9 - 3 Cycles 4 - 8 Basestations	15.82	10.85	10.80	12.58	14.04	15.82	12.95	12.11
Path 10 - 3 Cycles 4 Basestations	6.39	7.27	6.65	7.61	6.25	6.39	8.32	5.69
Path 11 - 3 Cycles 4 Basestations	6.99	8.54	6.69	7.99	6.97	6.99	8.13	6.37
Path 12 - 3 Cycles 4 Basestations	7.59	9.45	8.44	8.56	9.13	7.59	8.64	7.03
Path 13 - 3 Cycles 8 - 4 Basestations	17.64	12.00	11.77	14.03	14.35	17.64	14.96	12.37
Path 14 - 3 Cycles 8 Basestations	7.77	9.22	8.61	9.68	8.15	7.77	10.46	7.52
Path 15 - 3 Cycles 8 Basestations	8.71	10.34	9.00	10.23	9.05	8.71	9.81	8.43
Path 16 - 3 Cycles 8 Basestations	9.05	11.03	9.70	10.63	10.20	9.05	10.01	8.85
Path 17 - Random 4 - 8 Basestations	22.22	21.08	21.01	21.62	21.41	22.22	19.37	20.99
Path 18 - Random 4 Basestations	9.56	10.57	9.81	10.44	10.28	9.56	11.38	9.12
Path 19 - Random 8 - 4 Basestations	25.90	23.61	23.16	24.72	23.81	25.90	22.46	23.18
Path 20 - Random 8 Basestations	15.64	17.67	17.30	16.94	16.26	15.64	18.36	15.66
Average	13.92	13.81	13.36	14.01	14.22	13.92	14.21	12.93
Std	5.95	5.07	5.35	5.18	5.2	5.95	4.54	5.34

ditionally, robots operating in complex and dynamic domains will be required to make multiple decisions across a period of time. A forgetting algorithm's average performance along a path may frequently be more important than the best performance available at any one specific point in time.

The average estimation error generated by a forgetting algorithm or No Forgetting along a path was computed and is presented in Table VIII.1. Values in red represent average estimation error values worse than those generated by No Forgetting, for instance, Queue Static's average estimation error of 11.10 on path 2. Teal text indicates estimation values equal to No Forgetting performance, for instance, SIMPLE's average estimation error on path 1. The results from the 100 path instances for each path were averaged together. Analysis of the data at the path instance level would have been heavily influenced by the stochastic nature of the algorithms and noise in the environment. Path instance level analysis was not performed. The results were rounded to 2 decimal places. All data analysis and table coloring was performed on the rounded values to minimize the effects of noise. The standard deviation of the estimation values was also computed and those results are located in Table VIII.2.

Except for SIMPLE, which generated identical estimation error values as No Forgetting, every forgetting algorithm outperformed No Forgetting on some paths, but generated greater estimation error on others (Table VIII.1). The ActSimple algorithm resulted in an overall average estimation error of 12.93, the smallest across all forgetting algorithms and No Forgetting (Table VIII.1), but also generated a standard deviation of 9.12, the largest across all forgetting algorithms and No Forgetting (Table VIII.2). Applying ActSimple to the twenty paths generated less estimation error than No Forgetting on 13 paths, the most of any forget-

Table VIII.2: Along path standard deviation

Path	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 1 - 1 Cycle 4 - 8 Basestations	7.6	8.48	8.32	7.61	9.58	7.6	8.28	8.59
Path 2 - 1 Cycle 4 Basestations	8.42	8.32	8.37	8.22	7.4	8.42	7.58	8.41
Path 3 - 1 Cycle 4 Basestations	8.11	8.11	8.11	8.02	7.29	8.11	7.49	8.1
Path 4 - 1 Cycle 4 Basestations	8.09	8.09	8.09	8.03	7.06	8.09	7.4	8.09
Path 5 - 1 Cycle 8 - 4 Basestations	10.49	11.3	11.25	10.72	12.17	10.49	11.4	11.89
Path 6 - 1 Cycle 8 Basestations	13.15	12.93	13	12.84	11.86	13.15	11.83	13.12
Path 7 - 1 Cycle 8 Basestations	12.9	12.88	12.9	12.77	11.98	12.9	12.37	12.89
Path 8 - 1 Cycle 8 Basestations	13.02	13.02	13.02	12.91	12.04	13.02	12.5	13.02
Path 9 - 3 Cycles 4 - 8 Basestations	9.4	8.22	9.04	7.86	8.74	9.4	7.56	9.06
Path 10 - 3 Cycles 4 Basestations	5.89	5.56	5.79	5.44	6.55	5.89	5.46	6.2
Path 11 - 3 Cycles 4 Basestations	5.74	5.23	5.87	5.35	6.25	5.74	5.5	6.02
Path 12 - 3 Cycles 4 Basestations	5.73	5.28	5.67	5.37	5.74	5.73	5.51	5.98
Path 13 - 3 Cycles 8 - 4 Basestations	10.93	10.47	10.96	9.83	10.3	10.93	9.86	10.7
Path 14 - 3 Cycles 8 Basestations	9.04	8.54	8.75	8.38	9.45	9.04	8.45	9.15
Path 15 - 3 Cycles 8 Basestations	8.77	8.25	8.66	8.25	9.04	8.77	8.46	8.88
Path 16 - 3 Cycles 8 Basestations	8.77	8.15	8.56	8.23	8.67	8.77	8.49	8.85
Path 17 - Random 4 - 8 Basestations	6.91	7.39	7.63	6.63	7.08	6.91	7.22	6.89
Path 18 - Random 4 Basestations	6.21	5.65	6.04	5.67	5.98	6.21	5.54	6.48
Path 19 - Random 8 - 4 Basestations	8.76	9.73	9.84	8.75	8.94	8.76	9.88	9.25
Path 20 - Random 8 Basestations	10.97	9.36	9.64	9.92	10.5	10.97	8.81	10.89
Average	8.94	8.75	8.98	8.54	8.83	8.94	8.48	9.12
Std	2.36	2.42	2.32	2.38	2.14	2.36	2.25	2.31

ting algorithm (Table VIII.1). The ACT-R algorithm resulted in less estimation error than No Forgetting on the second most number of paths (8), but surprisingly generated the worst overall average estimation error (14.22). Unlike the ActSimple algorithm, when the ACT-R algorithm generated greater estimation error than No Forgetting, the difference was large ($\mu = 1.44$, $\sigma = 0.77$). On the paths where No Forgetting outperformed ActSimple, the average error difference was 0.02 ($\sigma = 0.01$). The Queue Dynamic and SIMPLE Update algorithms outperformed No Forgetting on 7 and 6 paths, respectively.

The statistical significance of the relative estimation error performance results from Table VIII.1 was evaluated with two Wilcoxon signed rank tests and are presented in Table VIII.3. These tests were performed in a similar fashion to those presented in Chapter V.1.3. The first test, labeled *Less than No Forgetting*, evaluated if the forgetting algorithms listed in the left-most column generated less average estimation error than No Forgetting. The second test, labeled *ActSimple less than*, evaluated if ActSimple resulted in less estimation error than No Forgetting and the other forgetting algorithms. Both tests were singled-sided and used an α of 0.05. Results that were found to be significant are indicated in bold and individual pairings that were not performed are marked with a —— symbol. The tests included the estimation error results from all 20 twenty paths.

The reduction in estimation error when compared to No Forgetting (column *Less than No Forgetting*) was significant for only ActSimple ($v = 28.0$, $p = 0.0014$). Results from the other forgetting algorithms were not significant. The second Wilcoxon signed rank test (column *ActSimple less than*) evaluated if ActSimple generated less estimation error. Results from No Forgetting and all of the forgetting algorithms were signifi-

Table VIII.3: Comparison of estimation error - Along Path Average

	Less than No Forgetting			ActSimple less than		
	n	v	p	n	v	p
No Forgetting	——	——	——	20	28.0	**0.0014**
Queue Static	20	128.0	0.8058	20	24.0	**0.0007**
Queue Dynamic	19	89.0	0.4144	20	43.0	**0.0096**
Random	20	134.0	0.8613	20	0.0	<**0.0001**
ACT-R	20	126.0	0.7848	20	1.0	<**0.0001**
SIMPLE	20	187.0	0.9995	20	28.0	**0.0014**
SIMPLE Update	20	115.0	0.6494	20	18.0	**0.0002**
ActSimple	20	28.0	**0.0014**	——	——	——

cant. No Forgetting ($v = 28.0$, $p = 0.0014$), Queue Static ($v = 24.0$, $p = 0.0007$), Queue Dynamic ($v = 43.0$, $p = 0.0096$), Random ($v = 0.0$, $p < 0.0001$), ACT-R ($v = 1.0$, $p < 0.0001$), SIMPLE ($v = 28.0$, $p = 0.0014$), and SIMPLE Update ($v = 18.0$, $p = 0.0002$) each resulted in more estimation error than ActSimple. These results suggest that ActSimple may be an effective and safe approach to improving system accuracy. No Forgetting and forgetting algorithms may not be able to realize a constant relative performance along the entirety of a path, but on average, ActSimple was able to generate less estimation error.

The effect of dynamism on the results was explored by using a single-sided Wilcoxon rank sum test to determine if No Forgetting and the forgetting algorithms generated less estimation error on static paths as compared to dynamic paths. Results from this test are presented in Table VIII.4, and was conducted in an identical fashion to the test presented in Chapter V.1.3. The test was performed with an α of 0.05, $n_A = 6$, and $n_B = 14$.

Table VIII.4: The effects of dynamism on estimation error - Along Path Average

	w	p
No Forgetting	0.0	<**0.0001**
Queue Static	12.0	**0.0059**
Queue Dynamic	12.0	**0.0059**
Random	8.0	**0.0017**
ACT-R	8.0	**0.0017**
SIMPLE	0.0	<**0.0001**
SIMPLE Update	9.0	**0.0023**
ActSimple	8.0	**0.0017**

The difference in average estimation error between static and dynamic paths was significant for No Forgetting and all of the forgetting algorithms, where each generated less estimation error on the static paths. These results suggest that dynamic conditions provide greater challenges to maintaining system accuracy than static conditions.

The presence of basestation configuration changes may alter the relative estimation error performance of No Forgetting and the forgetting algorithms. Four single-sided Wilcoxon signed rank tests were performed to evaluate the difference in estimation error that results from applying No Forgetting and the forgetting algorithms to static paths and dynamic paths. Results from these testes are presented in Table VIII.5. The first test (*Less Than No Forgetting - Static*) evaluates if the forgetting algorithms listed in the left-most column generated less estimation error than No Forgetting when only results from the fourteen static paths were considered. The second test (*Less than No Forgetting - Dynamic*) evaluates if the forgetting algorithms generated less estimation error than No Forgetting when only the results from the six dynamic paths were included. The third test (*ActSimple less than - Static*) evaluates if ActSimple results in less estimation error than No Forgetting and the other forgetting algorithms when applied to only the static paths. The fourth test (*ActSimple less than - Dynamic*) evaluates if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms when applied to only the dynamic paths. The α was 0.05 for each test.

Table VIII.5: Estimation error performance on static and dynamic paths - Along Path Average

	Less than No Forgetting						ActSimple less than					
	Static			Dynamic			Static			Dynamic		
	n	v	p	n	v	p	n	v	p	n	v	p
No Forgetting	——	——	——	——	——	——	14	28.0	0.0676	6	0.0	**0.0156**
Queue Static	14	105.0	1.0000	6	0.0	**0.0156**	14	6.0	**0.0009**	6	6.0	0.2188
Queue Dynamic	13	81.0	0.9960	6	0.0	**0.0156**	14	10.0	**0.0026**	6	9.0	0.4219
Random	14	105.0	1.0000	6	0.0	**0.0156**	14	0.0	**0.0001**	6	0.0	**0.0156**
ACT-R	14	102.0	0.9998	6	0.0	**0.0156**	14	0.0	**0.0001**	6	1.0	**0.0312**
SIMPLE	14	105.0	1.0000	6	12.0	0.6562	14	28.0	0.0676	6	0.0	**0.0156**
SIMPLE Update	14	105.0	1.0000	6	0.0	**0.0156**	14	0.0	**0.0001**	6	10.0	0.5000
ActSimple	14	28.0	0.0676	6	0.0	**0.0156**	——	——	——	——	——	——

The Wilcoxon signed rank test in Table VIII.5 for the *Less than No Forgetting - Static* condition evaluated if the forgetting algorithms generated less estimation error than No Forgetting on static paths. None of the results were significant, even though ActSimple ($\mu = 10.5862$, $\sigma = 3.5194$) was the only forgetting algorithm to average less estimation error on the static paths than No Forgetting ($\mu = 10.7950$, $\sigma = 3.2860$). The results of the Wilcoxon signed rank test for the *Less than No Forgetting - Dynamic* condition showed a different trend. The results from all of the forgetting algorithms, except SIMPLE, were significant. Queue Static ($v = 0.0$, $p = 0.0156$), Queue Dynamic ($v = 0.0$, $p = 0.0156$), Random ($v = 0.0$, $p = 0.0156$), ACT-R ($v = 0.0$, $p = 0.0156$), SIMPLE Update ($v = 0.0$, $p = 0.0156$), and ActSimple ($v = 0.0$, $p = 0.0156$) all generated less estimation error than No Forgetting when processing on the six dynamic paths. These results suggest that

Human-Inspired Forgetting based forgetting algorithms may be an effective approach to improving system accuracy in dynamic domains, and ActSimple may provide acceptable performance on static paths.

The Wilcoxon signed rank test for the *ActSimple less than - Static* condition evaluated if ActSimple generated less estimation error than No Forgetting and the other forgetting algorithms when only applied to static paths. Results from all of the forgetting algorithms, except SIMPLE were significant. Queue Static ($v = 6.0$, $p = 0.0009$), Queue Dynamic ($v = 10.0$, $p = 0.0026$), Random ($v = 0.0$, $p = 0.0001$), ACT-R ($v = 0.0$, $p = 0.0001$), and SIMPLE Update ($v = 0.0$, $p = 0.0001$) each generated more estimation error than ActSimple on static paths. Results from No Forgetting and SIMPLE were not significant. The Wilcoxon signed rank test for the *ActSimple less than - Dynamic* condition evaluated if ActSimple generated less estimation error on the dynamic paths. Results were significant for No Forgetting ($v = 0.0$, $p = 0.0156$), Random ($v = 0.0$, $p = 0.0156$), ACT-R ($v = 1.0$, $p = 0.0312$), and SIMPLE ($v = 0.0$, $p = 0.0156$). These results suggest that ActSimple is able to perform well across a variety of conditions, resulting in very good average performance. ActSimple may not be able to guarantee the best performance in every single tested condition, but the algorithm may be equipped to improve system accuracy across the diverse set of conditions potentially facing mobile robots.

The ActSimple algorithm generated the largest overall estimation standard deviation (Table VIII.2), although by itself, this metric provides little information. The standard deviation of a forgetting method's estimation error values describes the variability of the estimation values along the path, but does not indicate if this fluctuation is beneficial or detrimental. A forgetting method may consistently produce highly erroneous estimates, while another algorithm may oscillate from acceptable to highly accurate estimates. A third metric, signed squared deviation (SSD), was computed to analyze the effects of estimation variability. This metric is computed using Equation VIII.1 and compares an algorithm's performance against a reference trend. SSD is similar to the mean squared error except that the sign of the difference is maintained. Maintaining the sign allows the metric to compare the beneficial deviations from the reference to the negative deviations from the reference. The results in Table VIII.6 were computed with the average performance of all tested forgetting algorithms and No Forgetting acting as the reference. Values in red represent positive (detrimental) deviations and numbers in teal represent values equal to the reference.

$$SSD = \frac{\sum_{i=1}^{Length} \left[Sign(x_i - r_i)\,(x_i - r_i)^2 \right]}{Length} \tag{VIII.1}$$

The ActSimple algorithm was one of only three forgetting algorithms that generated a negative SSD value (-2.14) and was the only algorithm that produced negative SSD values for all paths (Table VIII.6). The Queue

Table VIII.6: Along path average signed squared deviation

Path	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 1 - 1 Cycle 4 - 8 Basestations	4.48	4.05	2.47	1.15	-0.54	4.48	-6.52	-4.54
Path 2 - 1 Cycle 4 Basestations	-0.83	-0.72	-0.79	-0.39	9.97	-0.83	5.58	-0.82
Path 3 - 1 Cycle 4 Basestations	-0.45	-0.45	-0.45	-0.31	7.33	-0.45	2.55	-0.45
Path 4 - 1 Cycle 4 Basestations	-0.51	-0.51	-0.51	-0.35	9.21	-0.51	2.67	-0.50
Path 5 - 1 Cycle 8 - 4 Basestations	6.15	0.83	0.79	1.52	-6.97	6.15	-2.19	-4.91
Path 6 - 1 Cycle 8 Basestations	-1.59	-0.85	-1.08	-0.48	12.08	-1.59	8.55	-1.50
Path 7 - 1 Cycle 8 Basestations	-0.56	-0.54	-0.55	-0.22	9.88	-0.56	1.12	-0.53
Path 8 - 1 Cycle 8 Basestations	-0.53	-0.53	-0.53	-0.24	9.80	-0.53	1.02	-0.52
Path 9 - 3 Cycles 4 - 8 Basestations	21.72	-14.51	-15.30	-0.97	4.47	21.72	-0.02	-3.94
Path 10 - 3 Cycles 4 Basestations	-0.39	0.34	-0.27	1.13	1.21	-0.39	4.66	-1.80
Path 11 - 3 Cycles 4 Basestations	-0.25	3.14	-0.60	0.84	1.19	-0.25	1.83	-1.39
Path 12 - 3 Cycles 4 Basestations	-1.00	3.46	1.05	0.17	4.24	-1.00	1.19	-3.00
Path 13 - 3 Cycles 8 - 4 Basestations	33.99	-14.81	-16.84	-0.52	1.23	33.99	3.44	-8.76
Path 14 - 3 Cycles 8 Basestations	-1.17	0.61	-0.27	1.95	1.97	-1.17	6.93	-1.80
Path 15 - 3 Cycles 8 Basestations	-0.50	2.58	-0.24	1.56	2.25	-0.50	0.86	-1.10
Path 16 - 3 Cycles 8 Basestations	-0.92	3.26	-0.11	1.24	3.25	-0.92	0.46	-1.55
Path 17 - Random 4 - 8 Basestations	4.35	-1.07	-2.31	0.74	0.69	4.35	-13.65	-0.18
Path 18 - Random 4 Basestations	-0.55	1.39	-0.23	0.33	0.14	-0.55	5.62	-1.66
Path 19 - Random 8 - 4 Basestations	10.27	-2.07	-4.07	1.33	-0.29	10.27	-8.59	-1.76
Path 20 - Random 8 Basestations	-2.65	3.38	1.90	0.15	-0.66	-2.65	6.11	-2.16
Average	3.45	-0.65	-1.90	0.43	3.52	3.45	1.08	-2.14
Std	9.14	5.12	5.04	0.86	4.82	9.14	5.43	2.06
Min	-2.65	-14.81	-16.84	-0.97	-6.97	-2.65	-13.65	-8.76
Max	33.99	4.05	2.47	1.95	12.08	33.99	8.55	-0.18

Dynamic algorithm generated the second largest number of negative SSD values with 16 and also produced the second most negative average SSD value (-1.90). SIMPLE and No Forgetting had the next highest number of individual path SSD values with 14; however, two of the positive values for each algorithm were at or above 21.72. These large values reflect the relatively poor performance of No Forgetting and the SIMPLE algorithm on the second half of the associated paths. The SIMPLE Update algorithm generated the second best overall estimation error at the end of the twenty paths (Chapter V), but only produced five negative individual path SSD values. The remaining fifteen paths all resulted in positive SSD values. The SIMPLE Update algorithm's overall average SSD value was 1.08, the fifth best overall average.

While the ActSimple algorithm had only the fourth best minimum SSD value (-8.76), ActSimple had the best and only negative maximum SSD value (-0.18). The minimum SSD represents an algorithm's best performance, while the maximum SSD represents an algorithm's worst performance. These results suggest that the ActSimple algorithm will not only be beneficial in some situations, but the algorithm may involve the least amount of risk concerning worst case estimation error.

ACT-R generated the largest average SSD value despite producing the third best average estimation error when applied to paths 1 - 20 and only end of path estimation error is considered (Chapter V). While SIMPLE and No Forgetting tied for the largest max SSD value (33.99), the common spurts of poor performance generated by ACT-R helped the method achieve the worst average SSD. Queue Dynamic produced the second best average SSD value despite the algorithm's fourth best end of path estimation performance (on paths 1

- 20). SIMPLE Update, which had the second best end of path average estimation error (on paths 1 - 20), produced an average SSD of 1.08.

The statistical significance of the SSD metric, when average performance was used as a reference, was tested with two single-sided Wilcoxon signed rank tests and the results are presented in Table VIII.7. The *Less than No Forgetting* test evaluated if the forgetting algorithms listed in the left-most column generated more negative SSD values than No Forgetting. The *ActSimple less than* test evaluated if ActSimple generated more negative SSD values than No Forgetting and the other forgetting algorithms. These tests used an α value of 0.05 and included all twenty tested paths.

Table VIII.7: Comparison of estimation error - Along Path SSD Average

	Less than No Forgetting			ActSimple less than		
	n	v	p	n	v	p
No Forgetting	——	——	——	20	28.0	**0.0014**
Queue Static	20	116.0	0.6629	20	60.0	**0.0487**
Queue Dynamic	19	84.0	0.3397	20	76.0	0.1471
Random	20	105.0	0.5073	20	0.0	<**0.0001**
ACT-R	20	123.0	0.7510	20	5.0	<**0.0001**
SIMPLE	20	159.0	0.9800	20	28.0	**0.0014**
SIMPLE Update	20	107.0	0.5364	20	38.0	**0.0053**
ActSimple	20	28.0	**0.0014**	——	——	——

The reduction in SSD values as compared to No Forgetting (column *Less Than No Forgetting*) was only significant for ActSimple ($v = 28.0$, $p = 0.0014$). The Wilcoxon signed rank test for the *ActSimple less than* condition evaluated if ActSimple generated more negative SSD values than No Forgetting and the other forgetting algorithms. Results from No Forgetting and all of the forgetting algorithms, except Queue Dynamic, were significant. No Forgetting ($v = 28.0$, $p = 0.0014$), Queue Static ($v = 60.0$, $p = 0.0487$, Random ($v = 0.0$, $p < 0.0001$), ACT-R ($v = 5.0$, $p < 0.0001$), SIMPLE ($v = 28.0$, $p = 0.0014$), and SIMPLE Update ($v = 38.0$, $p = 0.0053$) each resulted in SSD values greater than ActSimple. These results suggest that, in addition to averaging less estimation error than No Forgetting and the other forgetting algorithms along the tested paths (Table VIII.3), ActSimple may be able to minimize periods of relatively poor performance. While the results from comparing ActSimple and Queue Dynamic were not significant, ActSimple generated a maximum SSD value of -0.18 and Queue Dynamic produced a considerable larger maximum SSD value of 2.47 (Table VIII.3). Mobile robots operating in challenging domains need to achieve high levels of performance while also preventing periods of poor performance that may cause a task to fail. The results suggest ActSimple may be the safest tested Human-Inspired Forgetting algorithm.

Testing of the relative estimation error performance for No Forgetting and each forgetting algorithm on

Table VIII.8: Along path No Forgetting signed squared deviation

Path	No Forgetting	Queue Static	Queue Dynamic	Random	ACT-R	SIMPLE	SIMPLE Update	ActSimple
Path 1 - 1 Cycle 4 - 8 Basestations	0.00	-2.51	-3.58	-1.25	-12.67	0.00	-17.84	-16.13
Path 2 - 1 Cycle 4 Basestations	0.00	0.10	0.04	0.19	16.25	0.00	10.12	0.00
Path 3 - 1 Cycle 4 Basestations	0.00	0.00	0.00	0.05	11.25	0.00	4.80	0.00
Path 4 - 1 Cycle 4 Basestations	0.00	0.00	0.00	0.03	13.82	0.00	5.08	0.00
Path 5 - 1 Cycle 8 - 4 Basestations	0.00	-8.03	-7.61	-1.87	-25.35	0.00	-14.84	-20.86
Path 6 - 1 Cycle 8 Basestations	0.00	0.45	0.25	0.49	22.12	0.00	17.33	0.01
Path 7 - 1 Cycle 8 Basestations	0.00	0.02	0.00	0.15	15.09	0.00	3.21	0.00
Path 8 - 1 Cycle 8 Basestations	0.00	0.00	0.00	0.10	14.87	0.00	2.97	0.00
Path 9 - 3 Cycles 4 - 8 Basestations	0.00	-68.41	-71.81	-29.67	-15.08	0.00	-30.56	-39.67
Path 10 - 3 Cycles 4 Basestations	0.00	1.12	0.12	2.04	4.04	0.00	7.31	-0.93
Path 11 - 3 Cycles 4 Basestations	0.00	4.32	-0.16	1.53	3.05	0.00	3.20	-0.73
Path 12 - 3 Cycles 4 Basestations	0.00	7.33	4.13	1.53	8.76	0.00	3.12	-0.64
Path 13 - 3 Cycles 8 - 4 Basestations	0.00	-90.08	-95.54	-39.72	-39.72	0.00	-38.57	-70.99
Path 14 - 3 Cycles 8 Basestations	0.00	2.99	1.14	4.95	6.97	0.00	13.39	-0.18
Path 15 - 3 Cycles 8 Basestations	0.00	4.81	0.17	3.41	4.92	0.00	2.44	-0.18
Path 16 - 3 Cycles 8 Basestations	0.00	7.18	1.46	3.97	6.02	0.00	2.03	-0.11
Path 17 - Random 4 - 8 Basestations	0.00	-9.60	-12.70	-1.60	-3.32	0.00	-30.97	-4.85
Path 18 - Random 4 Basestations	0.00	3.08	0.21	1.45	1.08	0.00	9.29	-0.31
Path 19 - Random 8 - 4 Basestations	0.00	-20.08	-26.38	-4.32	-13.14	0.00	-37.06	-19.94
Path 20 - Random 8 Basestations	0.00	11.70	8.65	3.47	0.82	0.00	16.39	0.03
Average	0.00	-7.78	-10.08	-2.75	0.99	0.00	-3.46	-8.77
Std	0	25.6	26.43	11.26	15.4	0	17.98	18.05
Min	0.00	-90.08	-95.54	-39.72	-39.72	0.00	-38.57	-70.99
Max	0.00	11.70	8.65	4.95	22.12	0.00	17.33	0.03

static paths versus dynamic paths was not conducted. Unlike the average estimation error metric presented in Table VIII.1, the SSD metric is dependent on the relative performance of No Forgetting and the other forgetting algorithms.

Wilcoxon signed rank tests evaluating the relative estimation error performance between No Forgetting and the forgetting algorithms when applied to static paths and dynamic paths were conducted. The majority of the results are similar to the results involving the average estimation error metric and are not presented. The only change resulted from testing if ActSimple generated less estimation error than ACT-R on dynamic paths. Unlike in Table VIII.5, the results were not significant.

SSD values were also computed using No Forgetting as a reference and the results are presented in Table VIII.8. The SSD values for No Forgetting are all 0.00, since it was used as the reference. The SIMPLE algorithm's performance was always within the effects of noise and the algorithm's SSD values of 0.00 reflect that trend. The ActSimple algorithm did not generate a negative SSD value under all conditions, unlike in Table VIII.6. Along five paths, the ActSimple algorithm generated SSD values of 0.00 and for two paths, ActSimple generated positive SSD values. However, these two SSD values were very small (0.01 and 0.03). The other forgetting algorithms produced fewer individual path SSD values that were negative.

Queue Dynamic produced a better average SSD value (-10.08 compared to ActSimple's -8.77), but generated nine positive individual path SSD values. Queue Dynamic and Queue Static both produced better minimum SSD values (-95.54 and -90.08 respectively) than ActSimple (-70.99), but also had much large maximum SSD values (8.65 and 11.70 compared to 0.03). ActSimple only had the third best minimum SSD, but its maximum SSD was substantially better than the values for all forgetting algorithms except SIMPLE. No Forgetting generated very poor relative estimation error when the robot traversed the environment for three cycles and a basestation change occurred, as shown in Figures VIII.19 and VIII.20 (Section VIII.1). Table VIII.8 supports this finding with the SSD values for all forgetting algorithms (except SIMPLE) generating substantial negative SSD values for these two paths.

The ActSimple algorithm generated the least amount of average estimation error when performance was considered along each path (Table VIII.1). Table VIII.6 shows ActSimple generating the best SSD value when the average performance of all of the forgetting algorithms and No Forgetting is used as the reference. While ActSimple did not generate the best average SSD value when No Forgetting was used as a reference, Table VIII.8 shows the use of ActSimple resulting in the least positive max SSD value (excluding SIMPLE). These results suggest that at least for the paths and environment tested, ActSimple may not always generate the best possible performance, but on average, ActSimple will produce the best average error estimates while minimizing the risks associated with filtering data. Many robotic designs may be affected to a greater extent by average and worst case performance than best case performance. ActSimple has the potential to improve average robotic performance without greatly degrading worst case performance.

The statistical significance of the SSD values generated when No Forgetting was used as a reference was evaluated with two single-sided Wilcoxon signed rank tests and the results are presented in Table VIII.9. The tests were conducted in an identical fashion to the results presented in Table VIII.7. All twenty paths were used in the tests and the α was set to 0.05.

Table VIII.9: Comparison of estimation error - Along Path SSD No Forgetting

	Less than No Forgetting			ActSimple less than		
	n	v	p	n	v	p
No Forgetting	——	——	——	20	28.0	**0.0014**
Queue Static	19	100.0	0.5856	20	62.0	0.0570
Queue Dynamic	17	65.0	0.3056	20	78.0	0.1650
Random	20	124.0	0.7625	20	0.0	<**0.0001**
ACT-R	20	129.0	0.8159	20	6.0	<**0.0001**
SIMPLE	——	——	——	20	28.0	**0.0014**
SIMPLE Update	20	107.0	0.5364	20	37.0	**0.0047**
ActSimple	20	28.0	**0.0014**	——	——	——

The *Less than No Forgetting* test determined if the forgetting algorithms generated SSD values more negative than No Forgetting. This SSD metric was computed by using the estimation error performance of No Forgetting as a reference, and as a result the SSD values for No Forgetting are 0.0 for each path. This analysis effectively also tested if the forgetting algorithms generated SSD values less than 0.0. The results for ActSimple were significant ($v = 28.0$, $p = 0.0014$), but the results for the other forgetting algorithms were not significant. The *ActSimple less than* test determined if ActSimple generated more negative SSD values than No Forgetting and the other forgetting algorithms. Results from this analysis were significant for No Forgetting ($v = 28.0$, $p = 0.0014$), Random ($v = 0.0$, $p < 0.0001$), ACT-R ($v = 6.0$, $p < 0.0001$), SIMPLE ($v = 28.0$, $p = 0.0014$), and SIMPLE Update ($v = 37.0$, $p = 0.0047$). Testing with Queue Static and Queue Dynamic did not generate significant results. These tests suggest that ActSimple was the only forgetting algorithm able to consistently generate negative SSD values and average less estimation error than No Forgetting.

Wilcoxon signed rank tests evaluating the relative estimation error performance between No Forgetting and the forgetting algorithms when applied to static paths and dynamic paths were conducted. The results are similar to the results involving the SSD metric where average performance was used as a reference and are not presented.

VIII.3 Summary

The performance of each forgetting algorithm and No Forgetting varies across paths as well as along each path. This chapter explored the along the path performance of No Forgetting and the seven forgetting algorithms by applying each forgetting algorithm and No Forgetting to the same twenty paths (paths 1 - 20) used during optimization (Chapter V) and recording performance at each time step. Results from this experiment demonstrate the benefits and effectiveness of using the ActSimple algorithm to improve system accuracy throughout the entirety of a path.

The estimation error generated by No Forgetting and the forgetting algorithms generally decreased as the robot traveled through the environment, but the reduction in estimation error did not occur in a monotonic fashion. Even on static paths, short bursts of increased estimation error were observed with both No Forgetting and the forgetting algorithms. These estimation error variations were nearly identical for approximately the first 600 time steps on each path. The inability to maintain monotonically decreasing error did not result from limitations in the forgetting algorithms, but instead the estimation error volatility was a direct result of the interaction of the environment and task.

After approximately the first 600 time steps, the forgetting algorithms were provided with enough readings for estimation error performance to begin to diverge. ACT-R frequently generated a relatively large increase in estimation error that lasted for several hundred time steps, while SIMPLE Update frequently resulted in oscillating amounts of estimation error. ActSimple did not exhibit these drastic increases in estimation error.

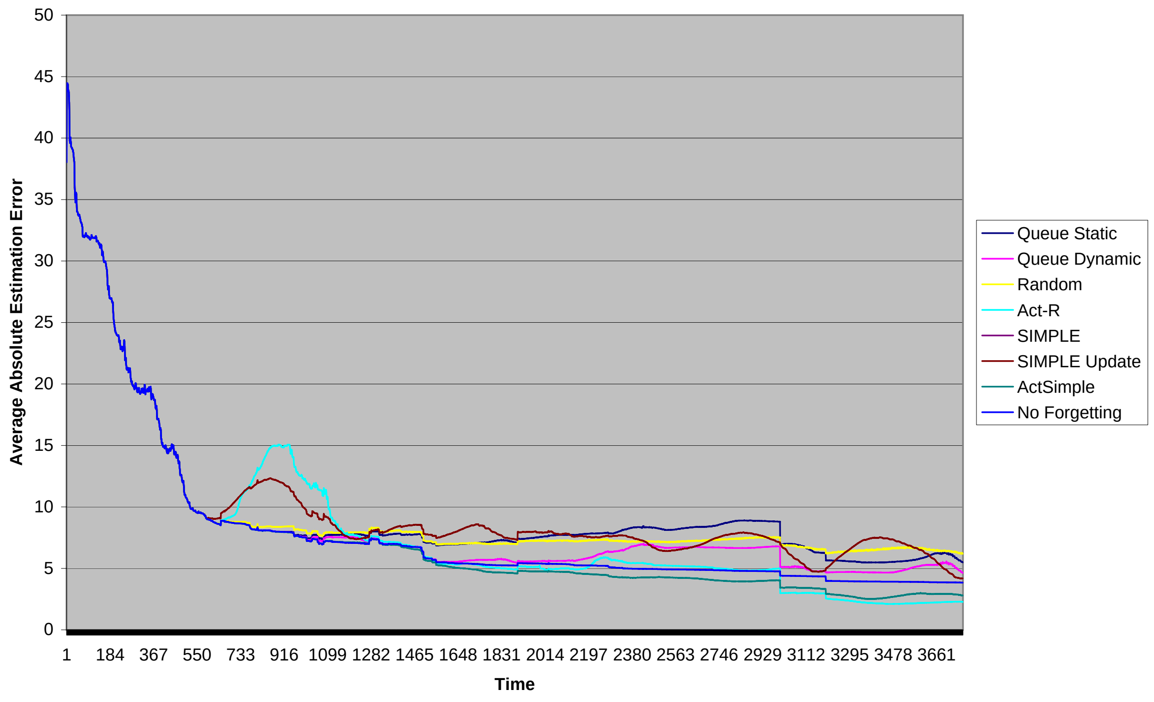


Figure VIII.27: Average estimation error on static paths and dynamic paths before the basestation configuration change

When basestation configuration changes occurred, the estimation error increased sharply for No Forgetting and each forgetting algorithm. The collection of additional readings again reduced the estimation error, but the relative performance of No Forgetting and the forgetting algorithms was more variable than at the beginning of each path. The ability for No Forgetting and the forgetting algorithms to reduce estimation error for both static paths and after a basestation configuration change were plotted in Figures VIII.27 and VIII.28. Figure VIII.27 shows the estimation error for No Forgetting and each forgetting algorithm averaged across the paths. Each path was included in the results, except that the portion of the dynamic paths that occurred after a basestation configuration change were not included. Figure VIII.28 shows the average estimation error generated by No Forgetting and the forgetting algorithms from the dynamic paths after the basestation configuration changes occurred. In this graph, time represents the amount of time since the basestation configuration change occurred.

Figures VIII.27 and VIII.28 both show the collection of additional WiFi signal strength readings generally resulting in the reduction of estimation error. However, the rate of estimation error reduction is not identical. After a basestation configuration change, both No Forgetting and the forgetting algorithms reduce estimation error at a slower rate and the relative performance between No Forgetting and the forgetting algorithms is more pronounced. No Forgetting and SIMPLE generally reduced estimation error at the slowest rate.

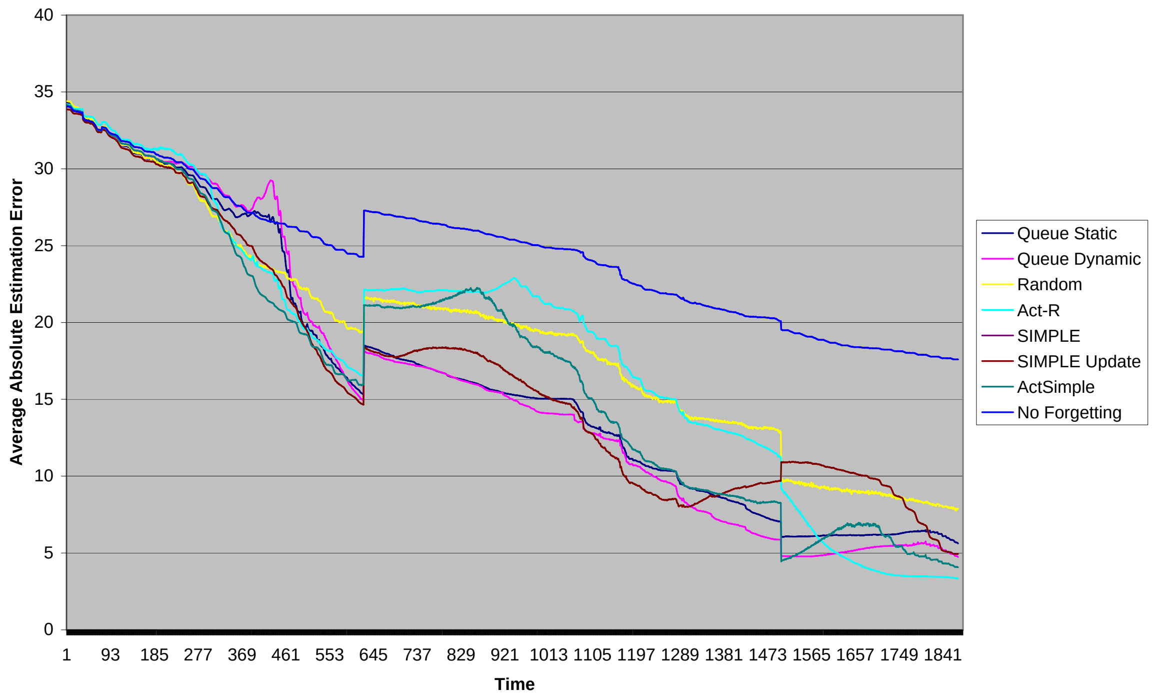


Figure VIII.28: Average estimation error on dynamic paths after a basestation configuration change

In addition to accuracy, algorithmic complexity can be an important factor when selecting and utilizing an algorithm. The average estimation error values from Table VIII.1 were contrasted with each algorithm's number of parameters and the results are presented in Figure VIII.29.

ActSimple possesses the largest number of parameters, but generated the smallest amount of estimation error for all twenty paths, the fourteen static paths, and the six dynamic paths. No Forgetting, which uses no parameters, produced the second smallest amount of estimation error on static paths, but resulted in the fourth best estimation error for all paths and the second worst estimation error on the dynamic paths. ACT-R and SIMPLE Update averaged greater amounts of estimation error than Queue Static and Queue Dynamic, despite possessing more parameters. While Queue Dynamic requires fewer parameters, the consistency of ActSimple's results suggest that the ActSimple algorithm may be an effective approach to improving system accuracy throughout tasks performed in both static and dynamic environments. The consistent performance of ActSimple may result from its unique combination of trace-based decay and similarity-based interference.

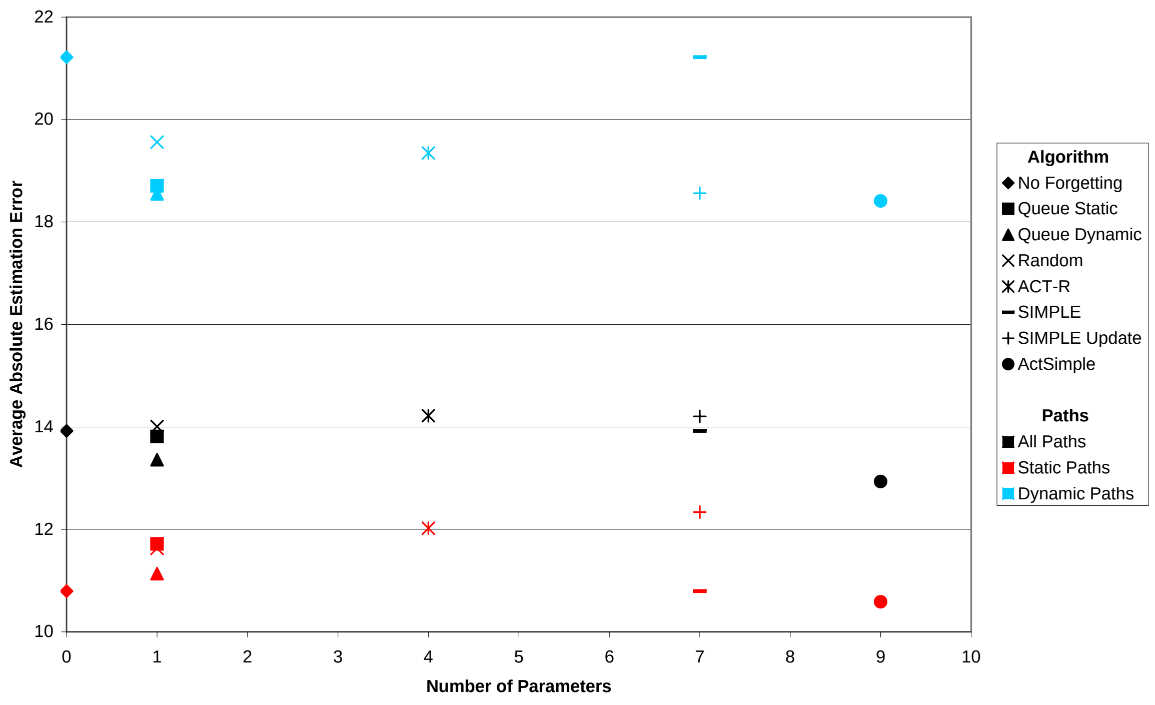


Figure VIII.29: Number of parameters vs. average estimation error along a path

CHAPTER IX

Pavlik and Anderson Modification

ActSimple possesses a large number of components, inspired by a diverse collection of cognitive architectures and models of human memory. A simplified version of the ActSimple algorithm was tested in the experiments presented in Chapters V - VIII. This simplification helped demonstrate many of the properties of ActSimple's two primary components, trace-based decay and similarity-based interference. This chapter presents an experiment that incorporates the Pavlik and Anderson modification (Pavlik and Anderson, 2003) into ActSimple's trace-based decay component.

IX.1 Optimizing the Pavlik and Anderson Modification

Pavlik and Anderson (2003) modified ACT-R's base level activation equation (Equation II.2 (repeated below)) to incorporate an item's activation level at each time of retrieval (Sims and Gray, 2004). The modified version of ACT-R's base level activation equation is shown in Equation IX.1. The $B_{i,j}$ term represents the activation present at the j^{th} perception or recall of the i^{th} item. This modification was applied to the ACT-R cognitive architecture to better realize the spacing effect of short-term memory (see Chapter II.4). The Pavlik and Anderson modification was incorporated into ActSimple using Equation IX.1, except that the B_i and $B_{i,j}$ terms were replaced with the b_i and $b_{i.j}$ terms. This implementation realized the full trace-based decay component within ActSimple, with the exception of the mental exertion component. The performance of the Pavlik and Anderson modification was evaluated when applied to both ActSimple and ACT-R. The Pavlik and Anderson modification versions of ACT-R (ACT-R PA) and ActSimple (ActSimple PA) were compared against the previously evaluated versions of the algorithms. No other changes were made.

$$B_i = \beta + ln(\sum_{k=1}^{n} t_k^{-d}) \tag{II.2}$$

$$\begin{aligned} d_{i,j} &= c_1 e^{B_{i,j}} + c_2 \\ B_i &= \beta + ln(\sum^{J_i} r_{i,j}^{-d_{i,j}}) \end{aligned} \tag{IX.1}$$

The addition of the Pavlik and Anderson modification required both forgetting algorithms to be fully re-optimized before performance comparisons were completed. Unlike the two stage optimization process performed in Chapter V, the ActSimple PA and ACT-R PA algorithms were optimized with the Coliny evolu-

1. Generate and evaluate an initial population.
2. Perform parent selection.
3. Apply crossover and mutation effects.
4. Evaluate the new individuals.
5. Select a new population.
6. Repeat.

Figure IX.1: Evolutionary algorithm process

tionary algorithm provided by the DAKOTA toolkit (DAKOTA, 2010). This algorithm required less computation and preliminary testing showed an equal effectiveness to the optimization approach taken in Chapter V. Figure IX.1 presents the steps taken by the evolutionary algorithm. An initial parameterization population is created and evaluated. Parent parameterizations are selected to form a set of new parameterizations. The crossover and mutation effects are applied to the selected individuals and the resultant parameterizations are evaluated. A new parameterization population is selected from the combination of both groups of parameterizations. The algorithm continues generating new populations until the results converge.

The optimization algorithm was configured to have a convergence tolerance of 0.0001 and to use a population size of 100. Initial seeding was performed randomly with the criteria that all instances were guaranteed to be unique. A two point crossover approach was taken with a crossover rate of 50%. The mutation rate was 100%, with mutations consisting of deviations generated from a zero mean normal distribution. A merit-based evaluation was employed and the probability of a parameterization being selected was proportional to its relative WiFi estimation error. During the selection phase, the best ten parameterizations from the combination of the two current groups were selected along with 90 randomly chosen parameterizations.

Environmental conditions were the same during the optimization described in Chapter V, except new path instances were used. Evaluations performed by the evolutionary algorithm employed ten path instances for each of the test paths. During the optimization process, DAKOTA stores all evaluated parameterizations and their respective estimation error values. Once the evolutionary algorithm completed, all parameterizations generating estimation error values under 6.5 were evaluated with 100 instances per path. Parameterizations for each algorithm were ordered based on the results of this second step. Evaluating the parameterizations with sets of 100 path instances permitted averaging out some of the effects of noise inherent in the environment and the algorithms.

The best parameterization for each Pavlik and Anderson algorithm is presented in Table IX.1, along with the previously discovered optimal parameterizations for both ACT-R and ActSimple. ACT-R and ActSimple

Table IX.1: Pavlik and Anderson parameterizations

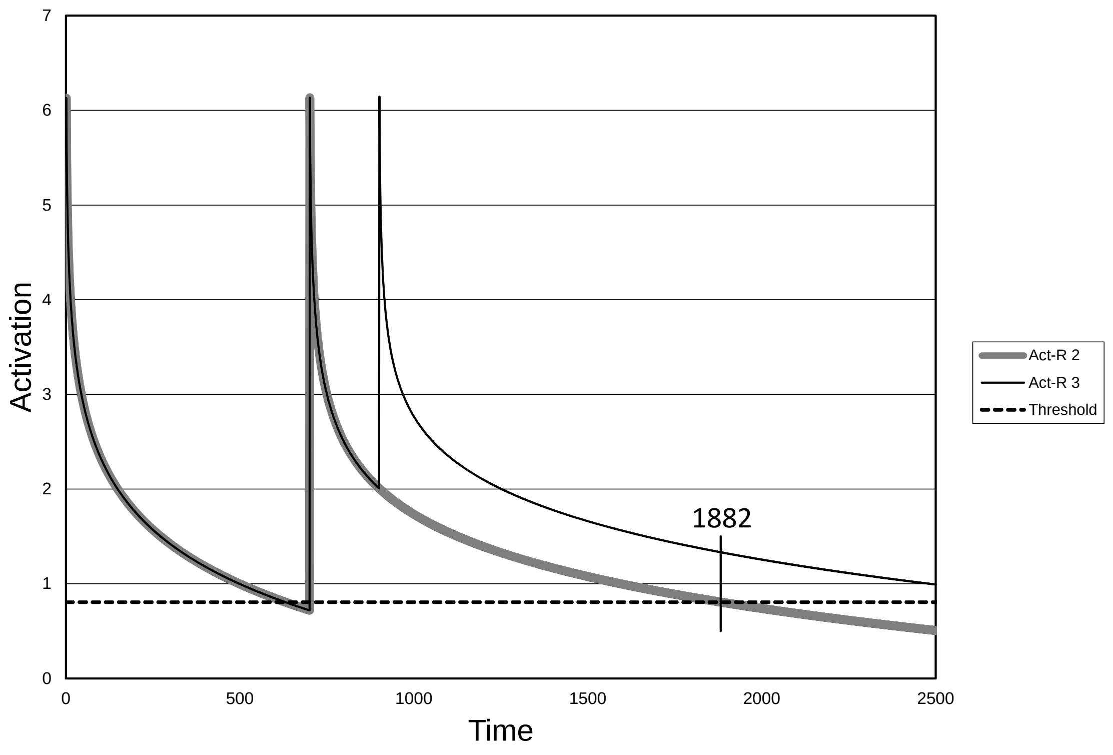
Algorithm	β	d	c_1	c_2	XYAttention	Strength Axis	c_3	c_8	ς	τ
ACT-R	6.1290	0.8257	—	—	—	—	—	—	104.6835	0.8065
ACT-R PA	-2.0181	—	8.5922	9.2281	—	—	—	—	80.9834	-68.8124
ActSimple	5.0000	0.9229	—	—	0.5000	0.0000	2.5000	0.1517	149.2500	0.0000
ActSimple PA	-2.1957	—	22.4574	18.7865	0.09093	0.8181	89.2226	0.6121	126.0423	-84.7130

parameter values vary significantly from the parameterizations calculated for the Pavlik and Anderson versions. In the original versions, β was 6.1290 and 5.0000 for ACT-R and ActSimple, respectively. β values for ACT-R PA and ActSimple PA were both approximately -2 (-2.0181 and -2.1957). The threshold values experienced a similar shift, changing from close to 0 (0.8065 and 0.0000) to large negative values (-68.8124 and -84.7130). Contrasted to the d values for ACT-R and ActSimple (0.8257 and 0.9229, respectively), which set the rate of forgetting present in the algorithm, the values c_1 and c_2 are large in magnitude (8.5922 and 9.2281 for ACT-R PA and 22.4574 and 18.7865 for ActSimple PA). In Equation IX.1, when $c_1 = 0$, $d_{i.j}$ is equivalent to c_2. As a result, setting $c_1 = 0$ and $c_2 = d$ provides identical behavior to ACT-R and ActSimple. The ActSimple specific parameters underwent similar changes. The weight provided to the strength axis in ActSimple transitioned from 0.0000 to 0.8181, while c_3 changed from 2.5000 to 89.2226. The c_8 parameter changed from 0.1517 to 0.6121, increasing the significance of the trace-based decay component.

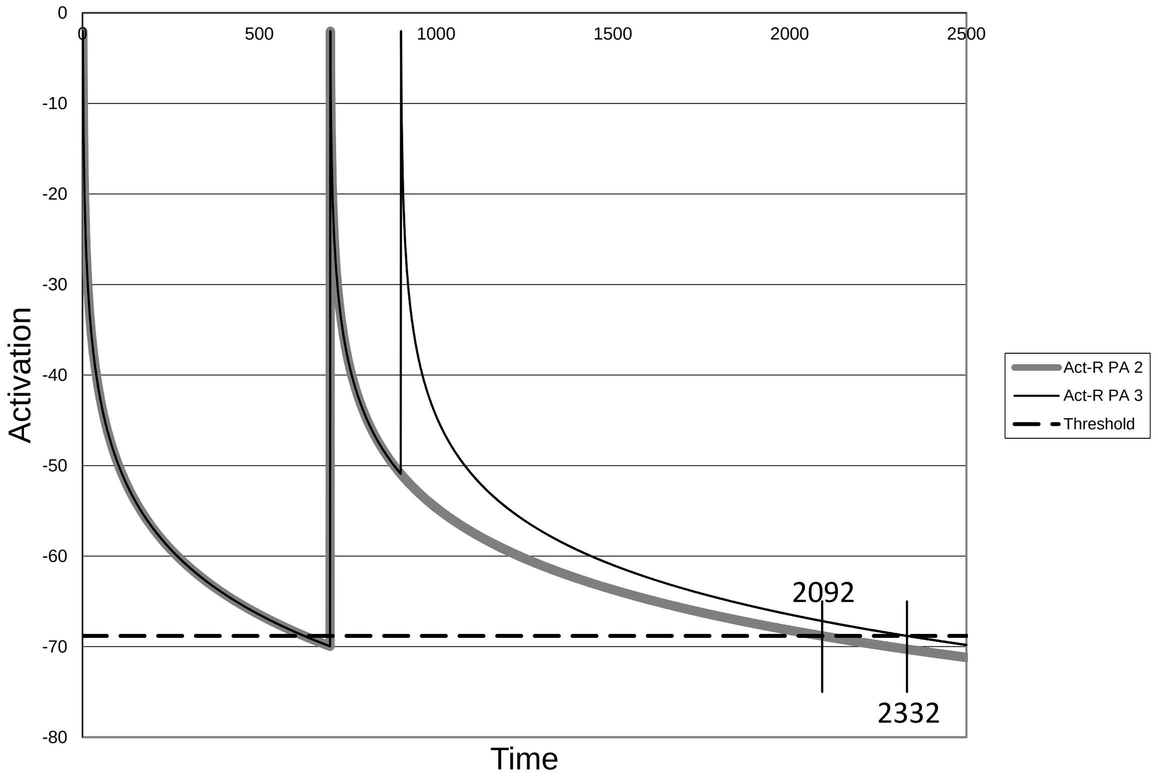
IX.2 Effects of the Pavlik and Anderson Modification

The Pavlik and Anderson modification alters the decay rate of items that have been reperceived or recalled. Figure IX.2 provides the changes to the rate of decay when the Pavlik and Anderson modification is applied to ACT-R. The figure was created with the recently determined optimal parameterization for ACT-R PA and all values were computed on integral values of time. Figure IX.2a presents the decay rates of ACT-R, while Figure IX.2b shows ACT-R PA's rate of forgetting. The ACT-R 2 and ACT-R PA 2 lines show a memory item that is initially perceived at time 0 and then reperceived at time 700. ACT-R 3 and ACT-R PA 3 display the same memory item except that it is reperceived twice, once at time 700 and again at time 900. Until the second reperception, ACT-R 2 is identical to ACT-R 3 and ACT-R PA 2 is identical to ACT-R PA 3. The horizontal lines represent threshold values, τ, for the forgetting algorithms. When the activation value crosses the threshold, the memory item will be recallable with a probability of 50%.

At time 1, the activation values for both algorithms start at the respective β values, but then the activation starts to decay. ACT-R PA 2 and ACT-R PA 3 cross their threshold at approximately time 627, while ACT-R 2 and ACT-R 3 cross slightly later, at roughly time 630. At time 700, the item is reperceived and the activation values sharply increased. The activation values for ACT-R 2 and ACT-R 3 at time 701 became



(a) ACT-R



(b) ACT-R PA

Figure IX.2: Comparison of the activation values generated by ACT-R and ACT-R PA

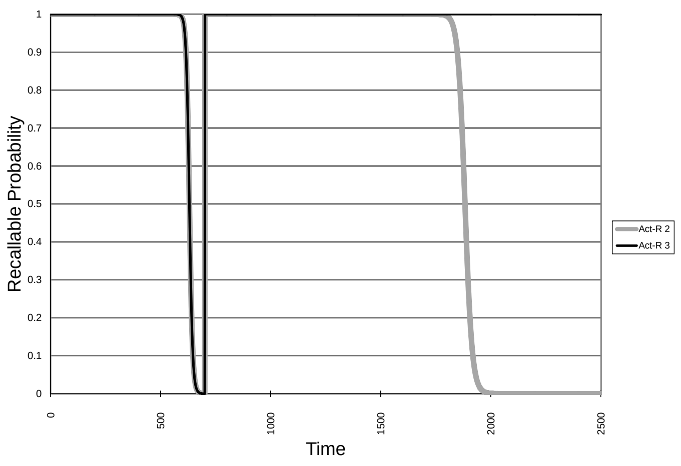
6.133, 0.004 larger than their β value, while ACT-R PA 2 and ACT-R PA 3's activations only return to approximately the β value. As time continued to progress, the activation values again began to decay. At time 900, the second reperception occurred and the activation values for ACT-R 3 and ACT-R PA 3 again increased sharply. The activation value for ACT-R 3 at time 901 became 6.145, while ACT-R PA 3 returned to -2.018. At approximately time 1882, ACT-R 2 crossed its threshold, while the ACT-R PA 2 activation did not cross until approximately time 2092. Unlike the initial crossings, ACT-R 2 crossed before ACT-R PA 2. At approximately time 2332, ACT-R PA 3 crossed its threshold value, while at time 2500, ACT-R 3 had an activation value of 0.992.

Activation values affect the probability that an item from memory will be made available to an existing robotic algorithm. Figure IX.3 presents the recall probabilities that are produced from the activation values in Figure IX.2. Figure IX.3a presents the recallable probabilities for ACT-R 2 and ACT-R 3, while Figure IX.3b shows the recallable probabilities for ACT-R PA 2 and ACT-R PA 3. The recall probabilities follow their respective activation values, maintaining a value of 50% when the activation crossed the respective threshold values. Until approximately time 1581, ACT-R 2 generated the same recallable probabilities as ACT-R 3, while ACT-R PA 2 produced identical recallable probabilities to ACT-R PA 3. Around time 1882, ACT-R 2 resulted in a recallability probability of 50%, while ACT-R 3 maintained a recallable probability of nearly 100% until at least time 2500. ACT-R PA 2 generated a recallable probability of 50% at approximeatly time 2092, while ACT-R PA 3 resulted in a recallable probability of 50% around time 2332. Figure IX.3c contrasts the recallable probability values of ACT-R 2 and ACT-R PA 2. The trends for these two conditions are the same, except ACT-R PA 2 generates quicker transitions from large probabilities to low probabilities.

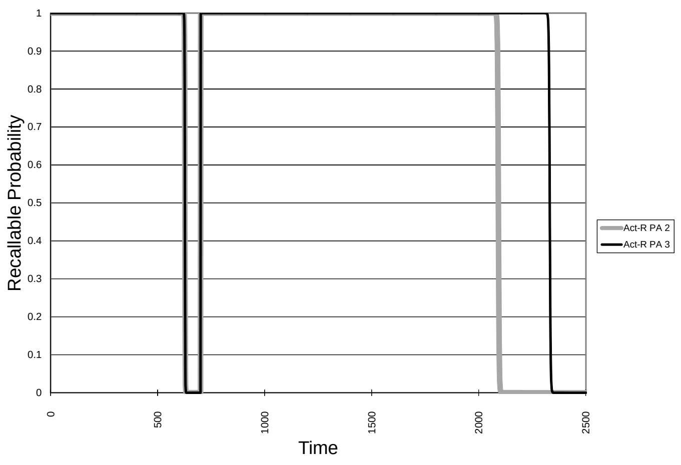
The data trends shown in Figures IX.2 and IX.3 confirm that the Pavlik and Anderson modification enables the ACT-R forgetting method to exhibit the spacing effect. This result can be seen when the memory item is reperceived the second time and the ACT-R PA 3 line decays appreciably faster than the ACT-R 3 line. Even in a dynamic environment, mobile robots may rapidly reperceive the same item for a short period of time. If the object represented by the memory item changes or disappears altogether, the increased rate of decay offered by the Pavlik and Anderson modification may enable robots to filter the memory item.

IX.3 Original Twenty Paths

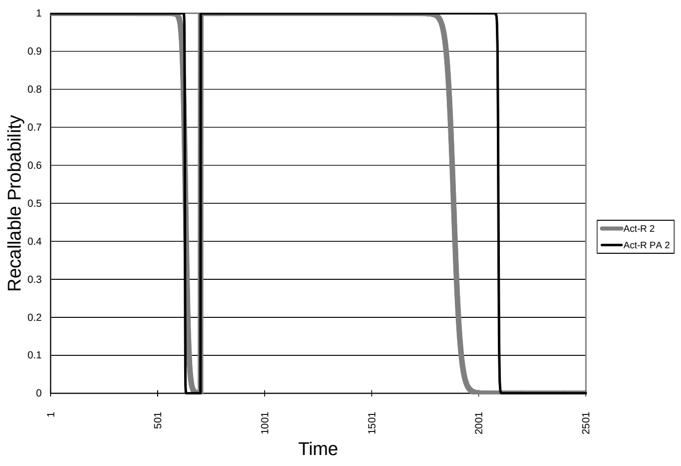
The ACT-R, ACT-R PA, ActSimple, and ActSimple PA algorithms were combined with Nearest Neighbor Interpolation with Perception Count Weighting to process 100 instances of paths 1 - 20 (Chapter IV.4). The resultant average recallable reading counts and average estimation error values for each of the four algorithms are presented in Table IX.2. ACT-R PA averaged 661.282 recallable items, 13.345 fewer than ACT-R's 674.627. Despite this reduction in recallable items, ACT-R PA achieved an average estimation error of 5.942,



(a) Comparison of the recall probabilities generated by ACT-R 2 and ACT-R 3



(b) Comparison of the recall probabilities generated by ACT-R PA 2 and ACT-R PA 3



(c)

Figure IX.3: Comparison of the recall probabilities generated by ACT-R and ACT-R PA

0.326 better than ACT-R's 6.268.

ActSimple and ActSimple PA followed a different trend. The Pavlik and Anderson version of the algorithm averaged 25.123 more recallable items and generated an average estimation value 0.472 larger than ActSimple. Interestingly, the performance of ACT-R PA and ActSimple PA was very similar, producing a recallable item difference of only 0.355 and an estimation error difference of 0.037. While ActSimple PA averaged more recallable items, the algorithm managed to maintain a slightly better estimation error than ACT-R PA.

Table IX.2: Pavlik and Anderson metrics

	ACT-R	ACT-R PA	ActSimple	ActSimple PA
Recall Average	674.627	661.282	636.514	661.637
Recall Std	83.32	98.377	53.001	101.757
Error Average	6.268	5.942	5.433	5.905
Error Std	3.368	2.758	2.588	2.733

The statistical significance of the relative average number of recallable readings and average estimation error performance results from Table IX.2 was tested with four single-sided Wilcoxon signed rank tests, with $\alpha = 0.05$. Results from these tests are presented in Table IX.3. The first test evaluated if ACT-R PA resulted in fewer average recallable readings than ACT-R. Results from this test were not significant. ActSimple was tested to determine if the algorithm averaged fewer recallable readings than ActSimple PA. This test was also not significant. Despite ACT-R PA averaging 13.345 fewer recallable readings than ACT-R and ActSimple averaging 25.123 fewer readings than ActSimple PA, the data does not indicate that the Pavlik and Anderson modification affects the average number of recallable readings.

Table IX.3: Effects of the Pavlik and Anderson modification

Test	n	v	p
Average number of recallable readings			
ACT-R PA less than ACT-R	20	91.0	0.3108
ActSimple less than ActSimple PA	20	81.0	0.1942
Average estimation error			
ACT-R less than ACT-R PA	20	79.0	0.1744
ActSimple less than ActSimple PA	20	91.0	0.3108

The effects of the Pavlik and Anderson modification on estimation error were also tested. ACT-R was tested to determine if the algorithm averaged less estimation error than ACT-R PA. The results were not significant. ActSimple was evaluated to determine if the algorithm resulted in less estimation error than ActSimple PA. Like ACT-R, the results were not significant. While the estimation error values from Table IX.2

maintained a 0.326 average estimation error difference between the ACT-R algorithms and a 0.472 average estimation error difference between the ActSimple algorithms, the results were not able to reveal an effect on estimation error by the Pavlik and Anderson modification.

Dynamism appears to have influenced the relative performance of the Pavlik and Anderson forgetting algorithms. Table IX.4 shows the performance differences in the ACT-R methods and the ActSimple Methods. The values in the table represent the change from the original method to the Pavlik and Anderson method, e.g., a negative error value indicates that the Pavlik and Anderson algorithm generated less error than the original algorithm.

Table IX.4: Effects of the Pavlik and Anderson modification along static and dynamic paths

	ACT-R, ACT-R PA Difference				ActSimple, ActSimple PA Difference			
	Recall		Error		Recall		Error	
	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static
Average	-34.25	-4.387	-2.099	0.434	64.903	8.076	-2.112	1.580
Std	53.671	21.708	3.207	1.196	102.337	100.762	2.451	2.306
Min	-114.55	-67.67	-6.477	-0.012	-33.75	-109.57	-6.185	-0.462
Max	6.57	12.2	0.072	4.556	202.4	173.34	-0.477	6.171

ACT-R PA averaged fewer recallable items for both the dynamic and static paths (-34.25 and -4.387 respectively), while the average estimation error only decreased in the presence of dynamism (-2.099). ACT-R PA produced larger estimation error (0.434) when the robot traveled static paths. ActSimple PA realized a similar error trend, decreasing error by 2.112 for dynamic paths, but increasing error by 1.580 for static paths. ActSimple PA averaged a greater number of recallable items under both conditions (64.903 for dynamic paths and 8.076 for static paths). Both ACT-R PA and ActSimple PA decreased average estimation error during dynamic paths by approximately the same amount (-2.099 and -2.112, respectively), but ActSimple PA's increase in error for static paths was 3.641 times as large as the change for ACT-R PA. The minimum and maximum error values show a similar trend between the ACT-R algorithms and the ActSimple methods. The PA methods reduced estimation error by at least 6.185 on one dynamic path, but generated at least 4.556 greater error on at least one static path. These extreme error deviations all occurred with paths containing a randomly generated trajectory.

The statistical significance of the average number of recallable readings and estimation error values from Table IX.4 was evaluated with eight single-sided Wilcoxon signed rank tests that are presented in Table IX.5. Significant results are indicated in bold and α was set to 0.05. ACT-R was evaluated to determine if the algorithm averaged fewer recallable readings than ACT-R PA on static paths, while ACT-R PA was tested to determine if the algorithm generated fewer recallable readings than ACT-R on dynamic paths. Results from both tests were not significant. ActSimple was tested to determine if the algorithm produced fewer recallable readings than ActSimple PA for static paths and for dynamic paths. Similar to the ACT-R tests, neither was significant. The data from applying the four algorithms to the set of twenty paths was unable to demonstrate the Pavlik and Anderson modification altering the average number of recallable readings.

Table IX.5: Statistical significance of the Pavlik and Anderson modification on static and dynamic paths

Test	Paths	n	v	p
Average number of recallable readings				
ACT-R less than ACT-R PA	Static	14	48.0	0.4039
ACT-R PA less than ACT-R	Dynamic	6	4.0	0.1094
ActSimple less than ActSimple PA	Static	14	47.0	0.3804
ActSimple less than ActSimple PA	Dynamic	6	5.0	0.1562
Average estimation error				
ACT-R less than ACT-R PA	Static	14	15.0	**0.0083**
ACT-R PA less than ACT-R	Dynamic	6	3.0	0.0781
ActSimple less than ActSimple PA	Static	14	16.0	**0.0101**
ActSimple PA less than ActSimple	Dynamic	6	0.0	**0.0156**

Wilcoxon signed rank tests were also used to determine if the Pavlik and Anderson modification had an effect of average estimation error. Testing if ACT-R resulted in less estimation error than ACT-R PA on static paths was significant ($v = 15.0$, $p = 0.0083$). ACT-R PA was tested to determine if the algorithm generated less estimation error than ACT-R when applied to dynamic paths, but the results were not significant. ActSimple's ability to reduce estimation error on static paths compared to ActSimple PA was found to be significant ($v = 16.0$, $p = 0.0101$). Significance was also found when testing if ActSimple PA resulted in less estimation error than ActSimple on dynamic paths ($v = 0.0$, $p = 0.0156$). The Pavlik and Anderson modification appears to increase estimation error when applied to static paths. However, the Pavlik and Anderson modification may reduce estimation error on dynamic paths. While the estimation error reduction from ACT-R PA was not significant when applied to the six dynamic paths, the average estimation error reduction shown in Table IX.4 was 2.099.

IX.4 Randomly Generated paths

The effects of random movement by the simulated robot were evaluated by applying the four forgetting algorithms to 500 new paths, all randomly generated. These paths were created based on twenty combinations of basestation configurations and path length (referred to as a path class). Half of the path classes were static, involving only the four basestation configuration or the eight basestation configuration, while the other half switched configurations at the path's midpoint. Path lengths ranged from 600 steps to 6000 steps. Twenty five paths were created for each path class and 100 instances were tested for each path. The random nature of each path allowed for large potential performance deviations between paths in each class.

The overall results of applying the four forgetting methods to the 500 randomly generated paths are shown in Table IX.6. Compared to the results from the original twenty paths (Table IX.2), all four forgetting methods averaged fewer recallable items. Out of the original set of twenty paths, sixteen did not involve a randomly generated path. These sixteen paths primarily involved the robot conducting a nearly complete investigation of the environment, repeatedly traveling from one side of the environment to the other. On average, the number of readings that are taken at each location within the environment will be more consistent than along the 500 new randomly generated paths. Frequently visiting the same locations reduces the total number of unique readings that can be collected, consequently reducing the number of readings that will be recallable.

Table IX.6: Pavlik and Anderson random path overall performance

	ACT-R	ACT-R PA	ActSimple	ActSimple PA
Recall Average	610.016	542.385	513.009	546.396
Recall Std	142.282	96.424	79.138	99.139
Error Average	12.449	12.31	11.756	12.307
Error Std	8.201	7.865	8.279	7.901

The average error for all four forgetting methods roughly doubled compared to the estimation error generated with the original twenty paths. Randomly generated paths frequently do not continuously and exhaustively explore the environment, like the non-random paths from the original set. In addition to not visiting some locations within the environment, randomly generated paths can also involve locations being visited a small number of times. Visiting locations a small number of times reduces the number of readings per location, limiting the ability for the Nearest Neighbor with Perception Count Weighting algorithm to average out the effects of noise.

The ACT-R PA algorithm averaged fewer recallable items (542.385) compared to ACT-R (610.016), as shown in Table IX.6, while ActSimple PA averaged more recallable items than ActSimple (546.396 and 513.009, respectively). These results are similar to those presented in Table IX.2. The average estimation error also followed this trend. ACT-R generated an average estimation error of 12.449 compared to ACT-R PA's estimation error of 12.31. ActSimple generated an average estimation error of 11.756, while ActSimple PA produced a value of 12.307. Again following the performance pattern from the original set of twenty paths, the results from ACT-R PA and ActSimple PA were very close, deviating in recall by 4.011 and average error by 0.003.

The statistical significance of the results from Table IX.6 was tested with four single-side Wilcoxon signed rank tests, which are presented in Table IX.7. Significant results are shown in bold and $\alpha = 0.05$. Values from all 500 paths were used.

Table IX.7: Effects of the Pavlik and Anderson modification on randomly generated paths

Test	n	v	p
Average number of recallable readings			
ACT-R PA less than ACT-R	496	118600.0	<**0.0001**
ActSimple less than ActSimple PA	500	24199.5	<**0.0001**
Average estimation error			
ACT-R less than ACT-R PA	491	58885.0	0.3158
ActSimple less than ActSimple PA	500	44141.5	<**0.0001**

The first Wilcoxon signed rank test evaluated if ACT-R PA resulted in fewer recallable readings than ACT-R and the results were significant ($v = 118600.0$, $p < 0.0001$). ActSimple was evaluated to determine if the algorithm generated fewer recallable readings than ActSimple PA and was also significant ($v = 24199.5$, $p < 0.0001$). These results indicate that when applied to randomly generated paths, the Pavlik and Anderson modification may aid the ACT-R algorithm in reducing the average number of recallable readings. Conversely, the modification may cause ActSimple to make additional readings recallable. Changes to the average number of recallable readings may influence the resultant average estimation error.

ACT-R was evaluated to determine if the algorithm generated less estimation error than ACT-R PA, but the results were not significant ($v = 58885.0$, $p = 0.3158$). Results from testing if ActSimple resulted in less estimation error than ActSimple PA were significant ($v = 44141.5$, $p < 0.0001$). When applied to both static and dynamic paths, the results suggest that the Pavlik and Anderson modification may not improve overall average estimation error.

The average recallable reading counts and estimation error values for each path class were computed and the performance differences between ACT-R and ACT-R PA are shown in Table IX.8, along with the performance differences between ActSimple and ActSimple PA. Negative values represent a Pavlik and Anderson algorithm generating either less error or fewer recallable readings. In the table, the path class labels include the basestation configurations present in the class and the number of three step movements taken by the robot. For example, the path class label "Random-B4-200" represents the set of paths that only involve the four basestation configuration and include 200 three step movements (a total of 600 steps). The label "Random-B4-100-B8-100" describes the set of paths where the robot travels through the four basestation configuration for 300 steps and then the eight basestation configuration for 300 steps. Three step movement patterns were used in the creation of the random paths to reduce the occurrence of the robot traveling back and forth to a very limited number of locations. Similar to Table IX.4, the values in Table IX.8 represent the change in performance from the original form of a forgetting algorithm to the Pavlik and Anderson version

Table IX.8: Pavlik and Anderson random path class performance

	ACT-R, ACT-R PA Difference		ActSimple, ActSimple PA Difference	
Path Class	Recall	Error	Recall	Error
Random-B4-200	0.038	0	4.288	0.006
Random-B8-200	0.062	-0.001	4.43	0
Random-B4-500	-7.144	0.273	-49.08	1.125
Random-B8-500	-7.14	0.218	-33.619	1.832
Random-B4-1000	-48.16	2.317	-4.287	2.881
Random-B8-1000	-47.717	3.094	39.508	3.586
Random-B4-1500	-86.369	2.62	21.168	2.775
Random-B8-1500	-77.702	3.857	89.992	2.998
Random-B4-2000	-107.998	2.491	49.784	1.899
Random-B8-2000	-115.704	4.192	106.413	2.734
Random-B4-100-B8-100	0.049	0.001	9.59	0.058
Random-B8-100-B4-100	0.054	0	8.911	0.058
Random-B4-250-B8-250	-9.023	-0.248	-7.802	-0.249
Random-B8-250-B4-250	-9.215	-0.173	-5.619	0.224
Random-B4-500-B8-500	-118.021	-4.861	18.367	-3.705
Random-B8-500-B4-500	-102.552	-4.324	12.042	-2.579
Random-B4-750-B8-750	-151.651	-4.124	90.677	-2.384
Random-B8-750-B4-750	-130.364	-2.678	75.472	0.08
Random-B4-1000-B8-1000	-172.387	-3.273	132.225	-1.322
Random-B8-1000-B4-1000	-161.675	-2.16	105.279	0.996
Average	-67.631	-0.139	33.387	0.551
Std	61.656	2.735	50.494	2.005

of the forgetting approach.

ACT-R PA generated greater error on static paths, but less error on dynamic paths, except for the four 600 step path classes. The average estimation error deviations during the 600 step paths were at most 0.001. ACT-R PA's change in error (compared to ACT-R) was dependent on the length of the path. During both the static and dynamic paths, the average error for the 1500 step paths was roughly a tenth of the magnitude of longer paths. ACT-R PA's largest performance decrement occurred with the Random-B8-2000 path class, while the biggest performance gain occurred with the path class Random-B4-500-B8-500.

Relative performance between ActSimple PA and ActSimple was more variable (Table IX.8). ActSimple PA generated inferior performance for each static path class (except for Random-B8-200 where the error difference was zero), but both performance gains and losses were present in the dynamic path results. ActSimple PA's best relative error value (-3.705) occurred with path class Random-B4-500-B8-500, but path class Random-B8-1000-B4-1000 resulted in a performance decrement of 0.996.

On average, ACT-R PA resulted in fewer recallable items (-67.631) and less error (-0.139), while ActSimple PA generated more recallable items (33.387) and more error (0.551). ACT-R PA's and ActSimple's best relative error occurred with the same path class, Random-B4-500-B8-500.

These results suggest that the benefits from the Pavlik and Anderson modification for dynamic paths may be dependent on time. Using the Pavlik and Anderson modification, readings that are repeatedly perceived within a short time period will quickly decay once a basestation configuration change occurs that invalidates those readings. As the amount of time between basestation configuration changes (or the end of the path) increases, erroneous readings will become unreadable and valid readings that have not been perceived for a period of time will also be filtered. While traversing randomly generated paths, nothing prevents the robot from traveling to an area in the environment, collecting a number of samples and not returning to that location. The Pavlik and Anderson modification, in this situation, may result in readings from this area quickly becoming unrecallable despite their accuracy and benefit.

A clear difference in performance between static and dynamic paths is shown in Table IX.8. The average effects of each path type (static and dynamic) are presented in Table IX.9. ACT-R PA generated a reduced number of recallable items for both dynamic and static paths (-85.478 and -49.783, respectively), but the change for static paths was roughly 42% less than the change for dynamic paths. ActSimple PA resulted in greater numbers of recallable items (43.914 for dynamic paths and 22.86 for static paths) and similar to ACT-R PA, the magnitude of change was larger for dynamic paths. ACT-R PA reduced error during dynamic paths (-2.184), but increased error during static paths (1.906). ActSimple PA generated a similar trend, an error reduction for dynamic paths (-0.882) and an increase in error for static paths (1.984). Unlike ACT-R PA, the magnitude of change for ActSimple PA was larger for static paths.

Table IX.9: Pavlik and Anderson random path effects of dynamism

	ACT-R, ACT-R PA Difference				ActSimple, ActSimple PA Difference			
	Recall		Error		Recall		Error	
	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static
Average	-85.478	-49.783	-2.184	1.906	43.914	22.86	-0.882	1.984
Std	72.568	45.293	1.951	1.641	51.583	49.755	1.533	1.256

The statistical significance of using the Pavlik and Anderson modification with static paths and dynamic paths was tested with a set of eight single-sided Wilcoxon signed rank tests, which are presented in Table IX.10. These tests are similar to those presented in Table IX.5.

The first two Wilcoxon signed rank tests evaluated if ACT-R PA resulted in fewer recallable readings with static paths and dynamic paths. Results from both tests were significant (v = 29700.0, $p < 0.0001$). Significance was also found when ActSimple was tested to determine if the algorithm resulted in fewer recallable readings than ActSimple PA for static paths (v = 8372.0, $p < 0.0001$) and dynamic paths (v =

Table IX.10: Statistical significance on static and dynamic randomly generated paths

Test	Paths	n	v	p
Average number of recallable readings				
ACT-R PA less than ACT-R	Static	248	29700.0	<**0.0001**
ACT-R PA less than ACT-R	Dynamic	250	29700.0	<**0.0001**
ActSimple less than ActSimple PA	Static	248	8372.0	<**0.0001**
ActSimple less than ActSimple PA	Dynamic	250	3943.0	<**0.0001**
Average estimation error				
ACT-R less than ACT-R PA	Static	247	1013.0	<**0.0001**
ACT-R PA less than ACT-R	Dynamic	250	26321.0	<**0.0001**
ActSimple less than ActSimple PA	Static	244	277.0	<**0.0001**
ActSimple PA less than ActSimple	Dynamic	250	21853.0	<**0.0001**

3943.0, $p < 0.0001$). Both sets of tests agree with the results from Table IX.7, the Pavlik and Anderson modification appears to decrease the number of recallable readings generated by ACT-R, but increase the number of recallable readings resulting from ActSimple.

The Pavlik and Anderson modification's affect on average estimation error was also evaluated with four Wilcoxon signed rank tests. ACT-R was evaluated to determine if the algorithm generated less estimation error than ACT-R PA on static paths and the results were significant ($v = 1013.0$, $p < 0.0001$). Significance was also observed when ACT-R PA was tested to determine if the algorithm resulted in less estimation error than ACT-R on dynamic paths ($v = 26321.0$, $p <= 0.0001$). These results help to explain why the test evaluating the affect of the Pavlik and Anderson modification when applied to all 500 paths (Table IX.7) was not significant. The modification may improve estimation error performance on dynamic paths, but degrade performance on static paths. A similar set of tests were performed on the ActSimple algorithms. ActSimple's ability to generate less estimation error than ActSimple PA on static paths was significant ($v = 277.0$, $p < 0.0001$) and ActSimple PA's ability to reduce estimation error compared to ActSimple on dynamic paths was also significant ($v = 21853.0$, $p < 0.0001$). These results suggest that the Pavlik and Anderson modification may possess the ability to improve system accuracy on randomly generated dynamic paths.

On average, ACT-R PA and ActSimple PA generated less error on dynamic paths, but increased error on static paths. The use and application of the Pavlik and Anderson modification to both ACT-R and ActSimple may only be applicable in certain domains and for particular tasks. Some robotic systems, such as a security robots, may constantly and exhaustively patrol a large environment. Incorporating the Pavlik and Anderson modification may be detrimental in those situations. However, other robots may only monitor an environment as a secondary task. In these cases, movement patterns may possess a greater degree of variability and when operating within a heavily dynamic environment, such as a busy office setting, benefit from the addition of the Pavlik and Anderson modification.

Both the ACT-R PA and the ActSimple PA algorithms were optimized with the same set of twenty original paths. Out of these paths, six possess a basestation configuration change and only four incorporated a randomly generated trajectory (two of the randomly generated paths were static and two were dynamic). Re-optimizing the Pavlik and Anderson algorithms with an increased percentage of random dynamic paths has the potential to increase the benefits afforded by the forgetting approaches during dynamic and complex paths, but at the expense of static and simpler paths.

IX.5 Summary

The Pavlik and Anderson modification was applied to the ACT-R and ActSimple forgetting algorithms, creating the ACT-R PA and ActSimple PA forgetting algorithms. These two new forgetting algorithms were optimized with an evolutionary algorithm and then tested with the original set of twenty paths and a set of 500 randomly generated paths. Results from this chapter suggest the Pavlik and Anderson modification may improve system accuracy on dynamic paths.

Incorporating the Pavlik and Anderson modification into ACT-R and ActSimple resulted in decreased estimation error when the forgetting algorithms were applied to dynamic paths. The decrease in estimation error between ActSimple PA and ActSimple was significant for both sets of dynamic paths. Accuracy improvements from ACT-R PA, compared to ACT-R, was significant when processing the set of randomly generated dynamic paths. However, applying the two Pavlik and Anderson modification forgetting algorithms to the sets of static paths resulted in increased estimation error. The Pavlik and Anderson modification did not have a significant effect on the average number of recallable readings when processing the original set of twenty paths, but the average number of recallable readings decreased when processing the set of randomly generated paths. The benefits the Pavlik and Anderson modification appear to be dependent on the properties of the original forgetting algorithm, task, and environment.

The relative benefits of the Pavlik and Anderson modification are affected by the combined changes to average estimation error and number of recallable readings. The interaction between these two metrics for the original set of twenty paths in shown in Figure IX.4. The ACT-R and ActSimple algorithms generated greater amounts of estimation error on the dynamic paths as compared to the static paths, while the Pavlik and Anderson modification algorithms resulted in decreased error on the dynamic paths. The ACT-R PA algorithm resulted in fewer recallable readings and less estimation error than ACT-R when applied to the dynamic paths and the full set of paths. However, while ActSimple PA resulted in less estimation error than ActSimple when applied to dynamic paths, the modification generated a greater number of recallable readings for all three conditions and increased estimation error for the static paths and the full set of twenty paths.

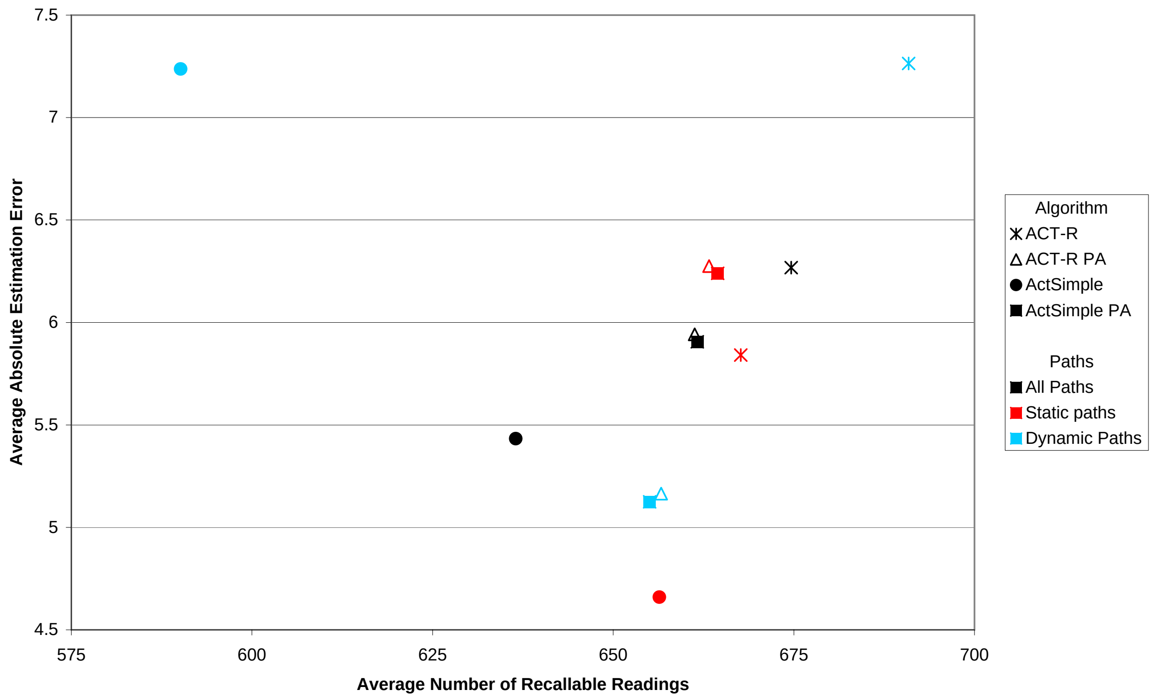


Figure IX.4: Average number of recallable readings vs. average estimation error for the original paths

A comparison of the average number of recallable readings and estimation error for the randomly generated paths is presented in Figure IX.5. The ActSimple algorithm combined the smallest number of recallable readings and the least amount of estimation error for the set of static paths and the full set of 500 random paths. Adding the Pavlik and Anderson modification resulted in additional recallable readings and greater amounts of estimation error. When applied to only the dynamic paths, the Pavlik and Anderson modification still resulted in more recallable readings, but decreased the amount of estimation error. Adding the Pavlik and Anderson modification to ACT-R decreased the number of recallable readings for all three sets of paths, but decreased the amount of estimation error for the set of dynamic paths and the full 500 paths.

The Pavlik and Anderson modification may improve system accuracy and performance under certain conditions. The degree to which dynamism is present in the environment and the computational cost of each additional recallable data item may determine the actual benefits of using the modification for any particular task or environment. Mobile robots operating in complex domains can experience a diverse array of varying conditions. These systems will require a forgetting algorithm capable of high levels of performance in all reasonable situations. However, some systems may operate within a constrained set of conditions or be capable of adapting forgetting algorithms as tasks and the environment change. In these situations, the use of the Pavlik and Anderson modification may result in increased system accuracy and performance.

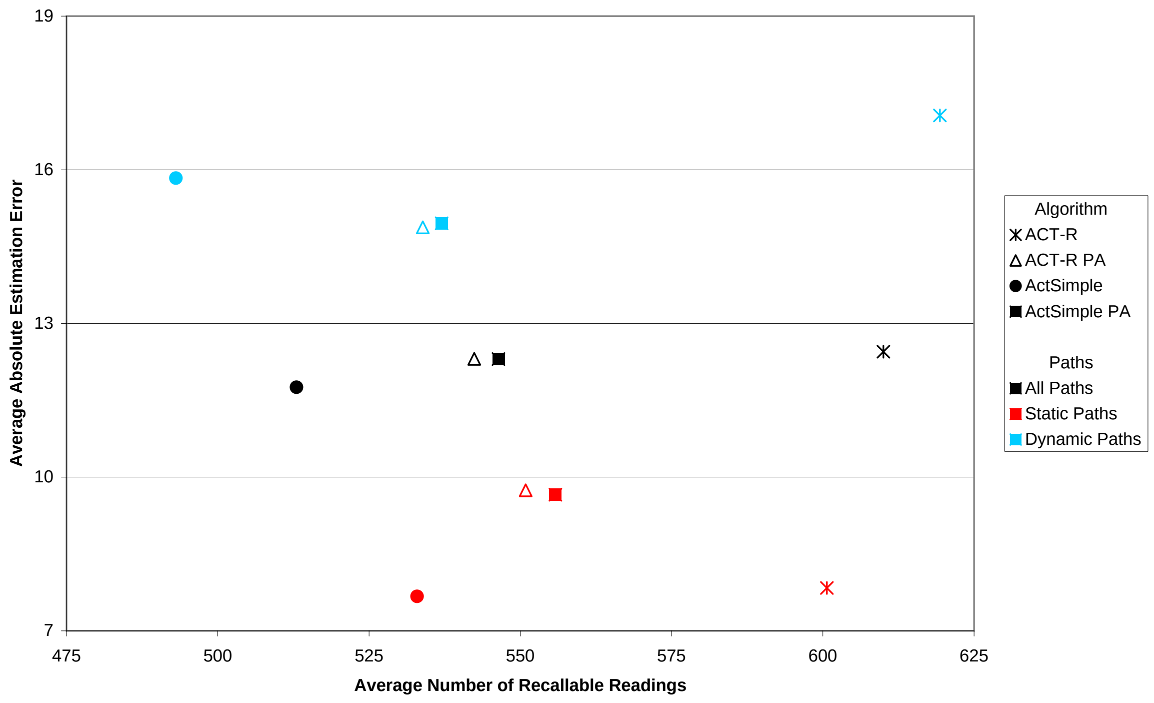


Figure IX.5: Average number of recallable readings vs. average estimation error for random paths

CHAPTER X

Summary of Contributions and Future Directions

This chapter reviews the work presented in this dissertation and summarizes the contributions from this research. An outline for potential avenues of future research are also presented.

X.1 Review

Perfect memory and recall provides a mixed blessing. While flawless recollection of episodic data and procedural rules allows for increased reasoning, photographic memory hinders a robot's ability to operate in real-time, highly dynamic environments. The absence of forgetting can result in memory being filled by a tremendous volume of data, increasing both search time and the probability of over-learning. Contemporary robots are already overrun with vast volumes of data requiring real-time processing and the problem will only increase. Before robots realize human-level intelligence, a means of classifying the importance of each acquired datum and forgetting unnecessary, erroneous, and expired data will be required.

This dissertation has developed Human-Inspired Forgetting, a means of incorporating forgetting capabilities into current and future robotic systems. This approach may enable robots to remove unnecessary, erroneous, and out-of-date information, while increasing the ability to reliably and rapidly recall critical cues necessary for successful task completion. Instead of selecting an item from memory to complete a task, Human-Inspired Forgetting filters the information presented to existing robotic algorithms. The pruned data may allow a diverse array of powerful, but task specific algorithms to realize improved accuracy while reducing cognitive load. The novel ActSimple forgetting algorithm was developed as an implementation of Human-Inspired Forgetting. This forgetting algorithm has been heavily inspired by a number of cognitive architectures, along with models of human memory and combines trace-based decay and encoding interference with belief state values, mental exertion, and output interference.

The benefits of the ActSimple algorithm and six other Human-Inspired Forgetting based forgetting algorithms were tested against the situation where No Forgetting was present. A WiFi signal strength estimation test domain was constructed to assess the ability of the forgetting algorithms to simultaneously reduce the amount of data presented to existing robotic algorithms and increase system accuracy. In the test domain, a robot explored an environment while collecting WiFi signal strength readings. At set points, the readings were used to generate estimated signal strength values for each location in the environment. During the experiments, two metrics were collected, the number of recallable readings presented to the existing robotic algorithm and the average estimation error at each location in the environment. The forgetting algorithms were

Table X.1: Reduction in average number of recallable readings

Experiment	No Forgetting average	3rd most average	3rd most reduction (%)	ActSimple average	ActSimple reduction (%)
Chapter V - Paths 1 - 20	1270.13	1025.51	19.259	636.958	49.851
Chapter V.3 - Paths 21 - 64	1090.794	897.408	17.729	600.877	44.914
Chapter VI - Real-World Data	848.316	692.467	18.372	338.887	60.052
Chapter VII - Other Noise Distributions	1303.568	797.987	38.784	601.359	53.868
Average	1128.202	853.343	23.536	544.520	52.171
Std	208.672	142.043	10.185	138.126	6.404

evaluated with a set of seven experiments, including the robot following a set of twenty paths (Chapter V.1.3), the use of alternate existing robotic algorithms (Chapter V.2), traveling a set of 44 paths (Chapter V.3), testing with WiFi signal strength readings collected from a real-world environment (Chapter VI), changing noise distributions in the environment (Chapter VII), testing performance along each path instead of just at the end (Chapter VIII), and adding the Pavlik and Anderson modification to the ActSimple and ACT-R forgetting algorithms (Chapter IX).

All forgetting algorithms, except for SIMPLE, reduced the average number of recallable readings, compared to No Forgetting, for each experiment where the metric was recorded. No Forgetting and the SIMPLE forgetting algorithm generated nearly identical results, consistently resulting in the largest number of recallable readings. In each experiment, the ActSimple algorithm filtered either the most or the second most number of readings. The reduction of average recallable reading count is presented in Table X.1. Results from ActSimple and the algorithm that generated the third largest number of recallable readings for each experiment are included. Each forgetting algorithm, except for SIMPLE, resulted in at least an average reduction of 23.536% of the number of recallable readings. ActSimple averaged a reduction of 52.171%. Mobile robots operating in real-world environments often have tight timing constraints and the reduction of data requiring cognitive processing may decrease computational load. However, system performance will only improve if filtering does not negatively affect system accuracy.

No Forgetting and the SIMPLE forgetting algorithm generated nearly identical estimation error values and averaged the largest amount of estimation error on the five experiments where performance was tested at the end of each path. The ActSimple algorithm averaged the least amount of estimation error in each experiment, except for the real-world experiment where the algorithm averaged the second smallest estimation error. The other forgetting algorithms resulted in more variable relative estimation error performance. The SIMPLE Update algorithm averaged the least amount of estimation error in the real-world experiment, but the third most when applied to paths 21 - 64. Use of the ACT-R forgetting method averaged the second smallest average estimation error with paths 21 - 64, but only the fifth best estimation error on the real-world data. The forgetting algorithms, except for SIMPLE, consistently resulted in less estimation error than No Forgetting, but only the ActSimple algorithm was able to maintain consistent relative estimation error reduction.

Relative estimation error performance of a forgetting algorithm may not be consistent throughout the entirety of a task. The along the path experiment tested the effectiveness of the forgetting algorithms to reduce estimation error at each point along the tested paths. The ActSimple algorithm resulted in the smallest average estimation error and the best average SSD value (using average performance as a reference), but No Forgetting did not generate the worst performance. Random forgetting, ACT-R, and SIMPLE Update each resulted in a greater amount of average estimation error and ACT-R produced the worst average SSD value. The forgetting algorithms were optimized by using the amount of estimation error generated at the end of each tested path. This optimization approach may have resulted in the reduced relative performance of some of the forgetting algorithms when estimation error was tested along each path. However, the tasks assigned to mobile robots are often difficult to perfectly model and often change slightly throughout task execution. The ActSimple algorithm was able to consistently maintain relative performance, by reducing estimation error, both at the end of paths and at each point along those paths.

Dynamism had an impact on the amount of estimation error generated by No Forgetting and the forgetting algorithms. Average estimation error increased when No Forgetting and the forgetting algorithms were applied to paths containing a basestation configuration change, but decreased when applied to static paths. The deviations between performance on static and dynamic paths was not consistent. No Forgetting and the SIMPLE forgetting algorithm resulted in the largest deviations between average estimation error on static paths and dynamic paths (Figures V.2, V.6, VI.3, VIII.29). Mobile robots assigned challenging tasks in complex environments require consistently high levels of performance. Results from this dissertation suggest that the use of Human-Inspired Forgetting may allow robotic systems to minimize performance fluctuations resulting from changing levels of dynamism present in the environment.

The ActSimple PA and ACT-R PA forgetting algorithms, which added the Pavlik and Anderson modification to ActSimple and ACT-R respectively, resulted in less estimation error when applied to dynamic paths from the original set of paths (paths 1 - 20). However, these two forgetting algorithms generated greater estimation error than ActSimple and ACT-R on the static paths. When ActSimple PA and ACT-R PA were applied to a large set of randomly generated paths, the algorithms again reduced estimation error on the dynamic paths but increase estimation error on the static paths. Under some conditions, forgetting algorithms may be capable of generating less estimation error on dynamic paths as compared to static paths, although overall accuracy and consistency may degrade.

Human-Inspired Forgetting has the potential to improve system accuracy, while decreasing the volume of data requiring potentially computationally expensive cognitive processing. The ActSimple forgetting algorithm provides an implementation of Human-Inspired Forgetting inspired by a diverse array of cognitive architectures and models of human memory. Experiments presented in this dissertation show that the ActSimple algorithm consistently reduced average estimation error, while simultaneously decreasing the average number of recallable readings provided to the existing robotic algorithms. ActSimple's consistency may result from the algorithm's unique combination of trace-based decay and similarity-based interference. Human-Inspired Forgetting, and the ActSimple algorithm in particular, has the potential to improve system accuracy, while minimizing the cognitive load experienced by real-time mobile robots. Forgetting algorithms may be unable to guarantee the best performance for every single condition, but may possess the ability to improve average performance, while limiting the severity and frequency of extreme performance degradations.

X.2 Summary of Contributions

Contributions of this dissertation include:

1. This dissertation has identified data management as a core area where non-domain specific algorithms and strategies can be developed to aid robotic systems in reducing cognitive load. These new algorithms and strategies may eventually assist robotic systems in improving their situational awareness. Understanding the current and future states of a complex and dynamic environment frequently requires many cues, data that can be easily overlooked or ignored. Improving robotic systems' ability to navigate large volumes of data may eventually lead to robots capitalizing on a greater percentage of these available cues, boosting performance and reliability.

2. This dissertation has put forth the idea that Human-Inspired Forgetting can be used as a means for robotic systems to filter incoming data streams and remove erroneous knowledge for the purpose of increasing data management capabilities and performance. Unlike some existing models of human forgetting and robotic systems, the presented approach does not select items from memory, but filters the data available to existing algorithms. Through this data reduction, traditional robotic algorithms may realize improved performance and increased accuracy.

3. ActSimple, a new algorithm designed to implement robotic forgetting was created. This algorithm incorporates a number of different components, each inspired by existing models and understanding of human forgetting and memory. The cognitive architecture ACT-R and the SIMPLE model of human memory provided the bulk of the inspiration for the algorithm and its name. ActSimple is a non-domain specific algorithm with a number of parameters and subcomponents that can be adjusted for particular

domains and tasks. These parameters can be tuned through either off-line or on-line learning mechanisms, minimizing design effort. A large infrastructure is not required by the ActSimple algorithm and multiple concurrent instances may be executed on the same system, each operating on different forms of data or sections of the robot.

4. Within the context of a WiFi signal strength estimation task, this dissertation presented initial exploration into the application of Human-Inspired Forgetting. Factors affecting robotic performance, including changes to task and the environment were investigated, revealing how forgetting capabilities can mitigate some of the challenges posed by complex and dynamic environments.

X.3 Future Directions

Many avenues exist for future work in Human-Inspired Forgetting for robotic systems. Additional components within ActSimple can be explored or added, Human-Inspired Forgetting can be applied to new forms of data and knowledge, and many robotic domains exist that may benefit from robotic forgetting capabilities.

The ActSimple algorithm combines a number of diverse components inspired by many models of human memory and cognitive architectures. This dissertation explored the components of trace-based decay, similarity-based interference, and the Pavlik and Anderson modification. Future experiments may investigate the benefits afforded by belief state values, mental exertion, and output interference.

A number of additional components may also be added to the ActSimple algorithm. Rehearsal forms a critical component in many trace-based decay theories of forgetting, see Chapter II.4.1, and Figure II.12 in particular. This cognitive strategy is analogous to a human rapidly and repeatedly stating a fact to prevent forgetting and may aid a robot in actively shaping the knowledge that is remembered and forgotten.

Many items encountered by robots possess a number of characteristics that allow for grouping. Color, size, heat, and location are all properties that may be important in robotic domains. Presently, the ActSimple algorithm treats all data equally. The introduction of a mechanism into ActSimple that "highlights" items within memory that fit a currently desired parameter set may be explored. Minsky (1986) proposed the idea of K-Lines, mechanisms within human memory that link relevant memories together. When a K-Line is "activated", all of the memories attached to the K-Line also become "activated" or easier to recall. A system that "highlights" memory items may provide a robot with an improved ability to utilize cues in recalling critical information.

Forgetting algorithms, and ActSimple in particular, possess a large number of independent parameters. This dissertation explored the effectiveness of off-line parameter space searches, but on-line implementations may be explored. While some reliable real-time metric will be required to ascribe a utility to individual parameter settings, nothing prevents the parameters used by ActSimple from being manipulated while a robot traverses its environment. Many domains where robots are and will be operating are fluid and dynamic. Optimal parameter settings may also fluctuate and on-line tuning may be one mechanism that allows ActSimple to adapt to these changes.

Many task, domain, and environmental factors can influence the absolute and relative performance of a forgetting algorithm and its parameterization. This dissertation created parameterizations for ActSimple and six other forgetting algorithms by optimizing performance with twenty paths. An analysis may be conducted to determine the relative benefits of creating generalized parameterizations, parameter values tuned for high levels of performance across diverse conditions, and specialized parameterizations, parameter values tuned for specific subsets of task and environmental conditions. In some situations, the use of a single general use parameterization may result in acceptable performance. However, some domains may allow for accurate sensing of environmental conditions and performance may increase if the robot rapidly switches between specialized sets of parameterizations.

Human-Inspired Forgetting may benefit a number of robotics domains. Many traditional robotic domains, such as Human-Robot Interaction (HRI) (Goodrich and Schultz, 2007) and disaster rescue (Murphy et al., 2000, 2008), may directly benefit from the introduction of robots with forgetting capabilities. Human-robot interaction involves humans and robots working cooperatively to solve common goals and in some cases, providing mutual assistance. Robots may one day become ubiquitous elements of human daily life, but they will require the ability to communicate naturally with humans. Along with the ability to provide human partners with the small cues and mannerisms that are common in human communication, robots will need to detect and correctly interpret cues provided by their human partners. Gaze (Mutlu et al., 2009), timing (Okuno et al., 2009), and body position (Yamaoka et al., 2009) can all influence the quality of communication and the quantity of information that is reliably passed between partners. Effects of cues and emotive behavior are not consistent. If robots are to one day be able to achieve natural interaction capabilities with humans, the ability to recall important cues, while ignoring inconsequential actions may be critical. Human-Inspired Forgetting may provide this capability. Through the interaction with an individual, a robot may be able to record communication cues and adapt its own behavior appropriately. With the inclusion of rehearsal mechanisms, robots may be well equipped to recall common cues and critical rare human reactions. These rare reactions may result from injuries or from a human becoming exceptionally angry.

Wilderness Search and Rescue (Adams et al., 2009), Urban Search and Rescue (Murphy et al., 2000), and future CBRNE deployments (Humphrey and Adams, 2009b) are all domains where robots are frequently tasked with searching an environment, detecting abnormalities, and identifying items that appear out-of-place. Unexpected items may signify humans trapped under rubble or explosive devices that have been concealed in a building. Forgetting may be able to improve the performance of robotic systems operating within these domains. Similar to the experiments presented in this dissertation (Chapters V - IX), forgetting may enable robots to gain an understanding of the current status of an environment and adapt to changes. These domains are often highly dynamic and continually evolve. Supplies are transported throughout the environment, rubble can shift, first responders setup and break down equipment, and victims may wander throughout the affected areas. Forgetting may allow robots operating under these conditions to remove outdated information, while still retaining salient and pertinent data.

Appendix A

Supporting Media for Chapter VIII

This appendix presents additional supporting media for Chapter VIII.

A.1 Additional Along the Path Performance Graphs

This section presents eight additional along the path performance graphs that were not included in Chapter VIII. Graphs for paths 3, 6, 7, 11, 14, 15, 16, and 19 are provided.

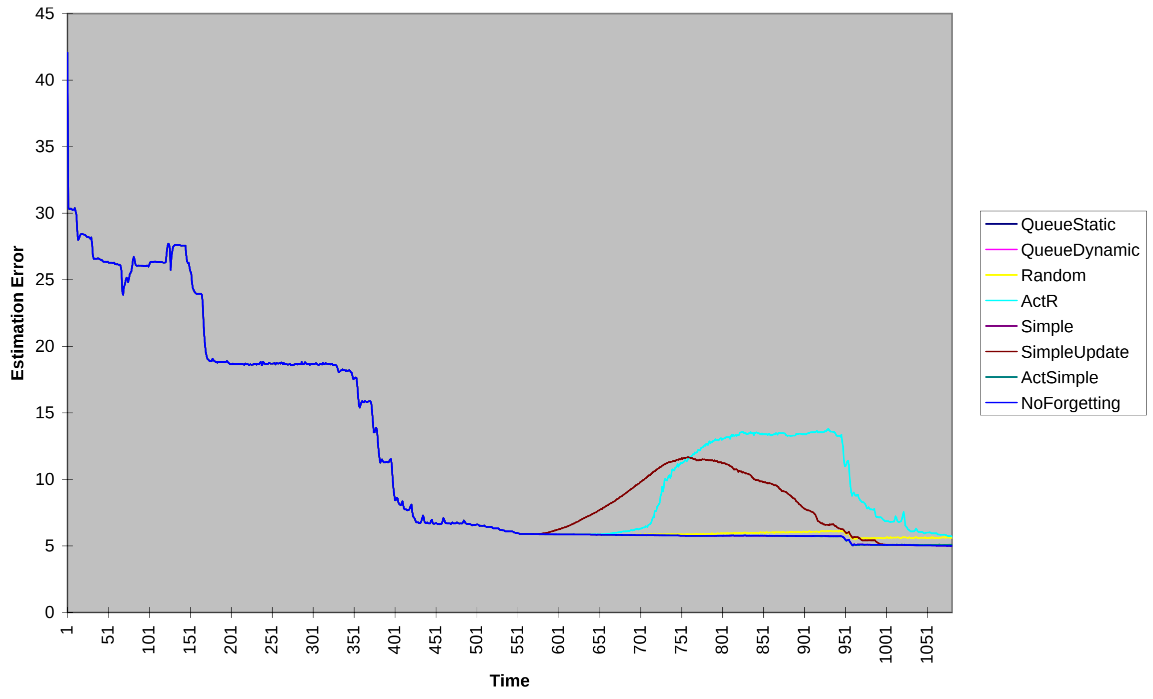


Figure A.1: Along path results for path 3 - 1 cycle 4 basestations

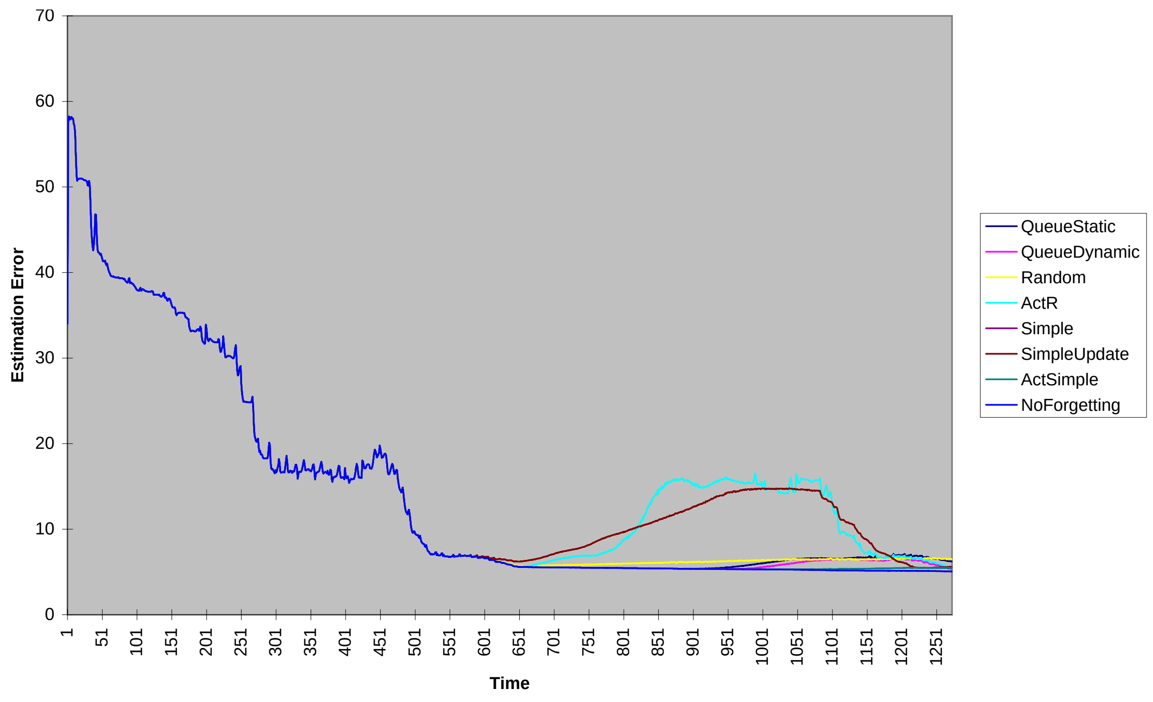


Figure A.2: Along path results for path 6 - 1 cycle 8 basestations

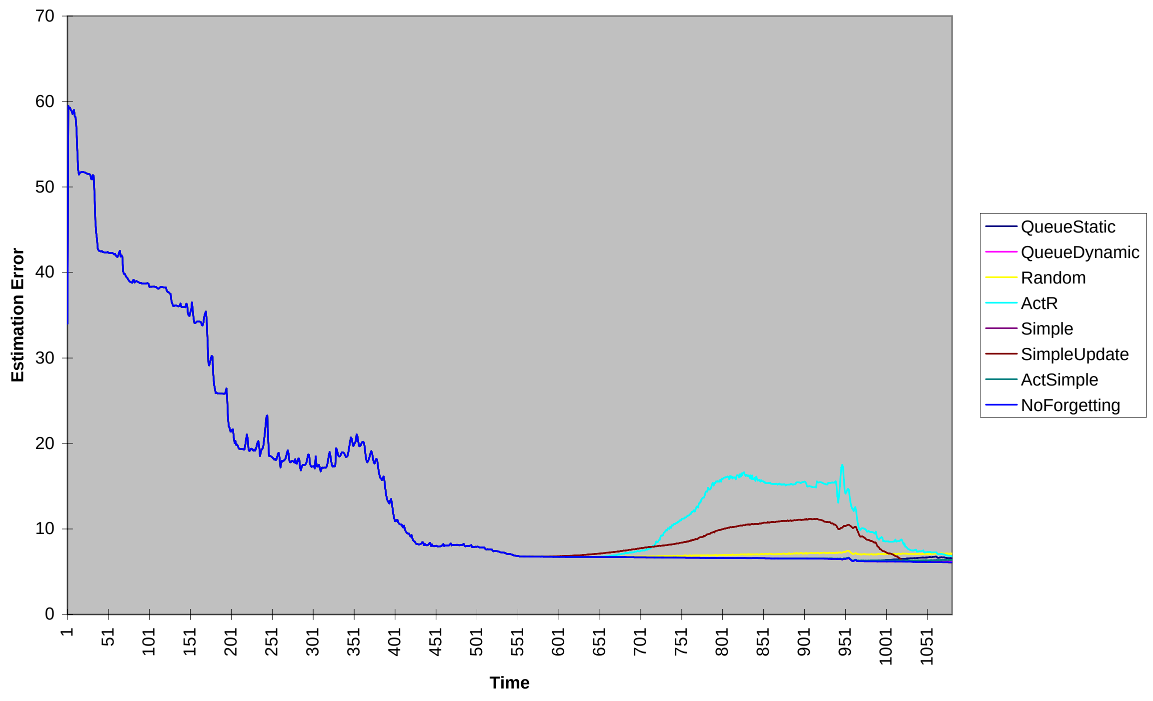


Figure A.3: Along path results for path 7 - 1 cycle 8 basestations

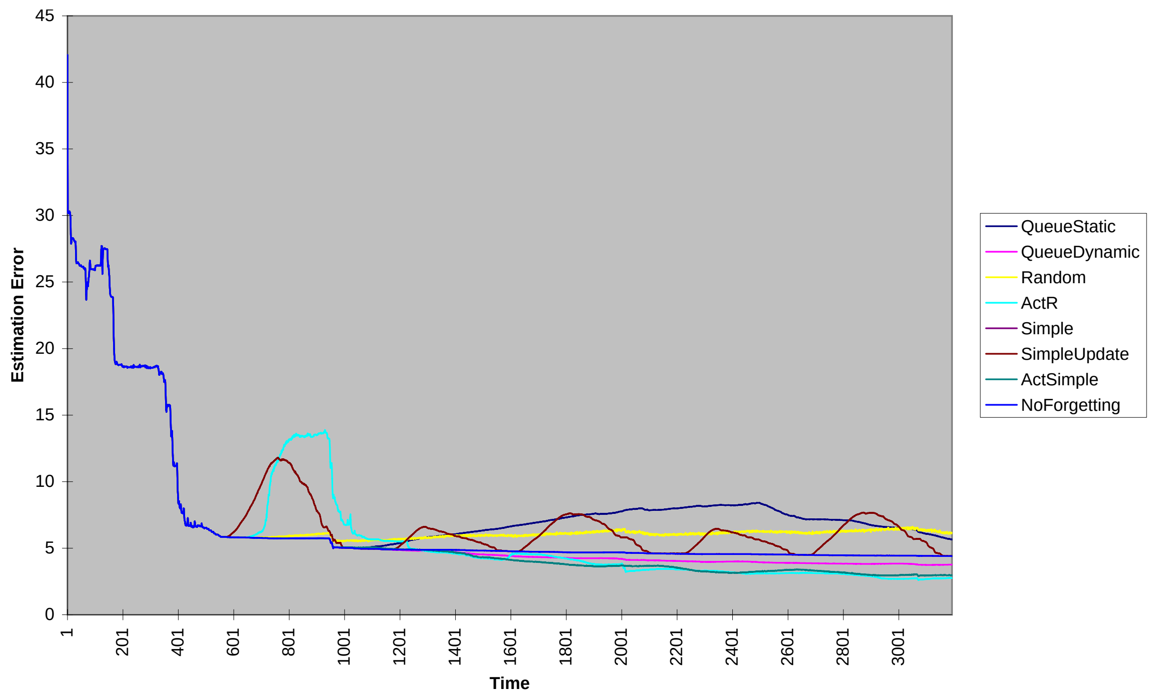


Figure A.4: Along path results for path 11 - 3 cycles 4 basestations

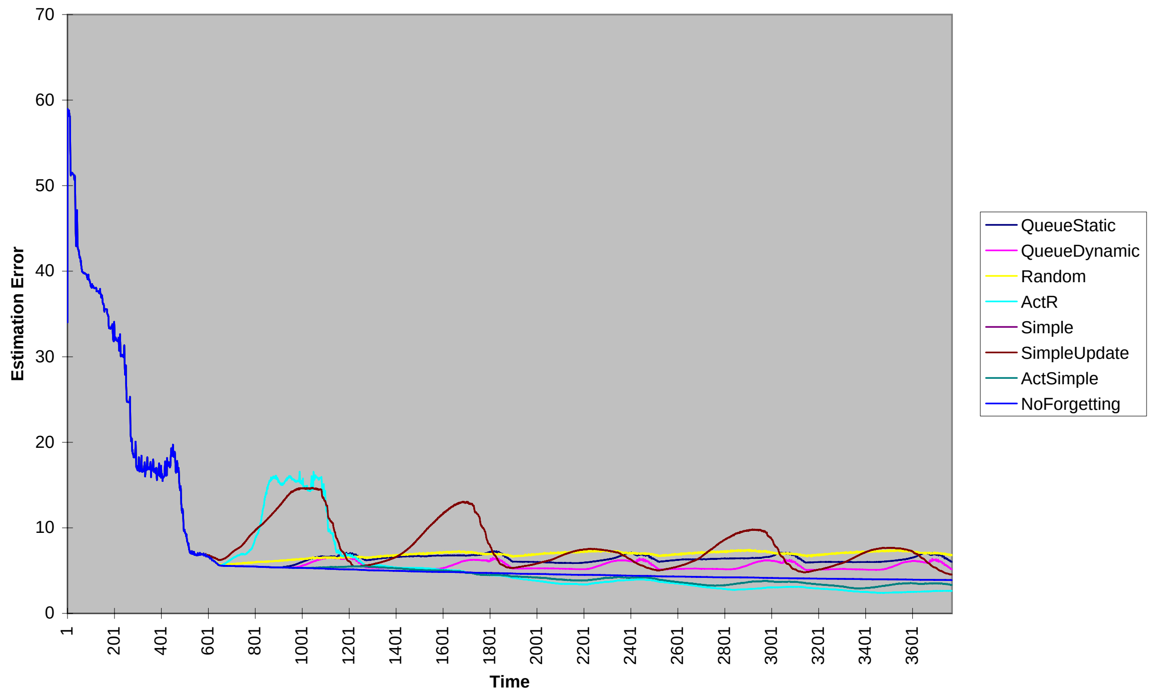


Figure A.5: Along path results for path 14 - 3 cycles 8 basestations

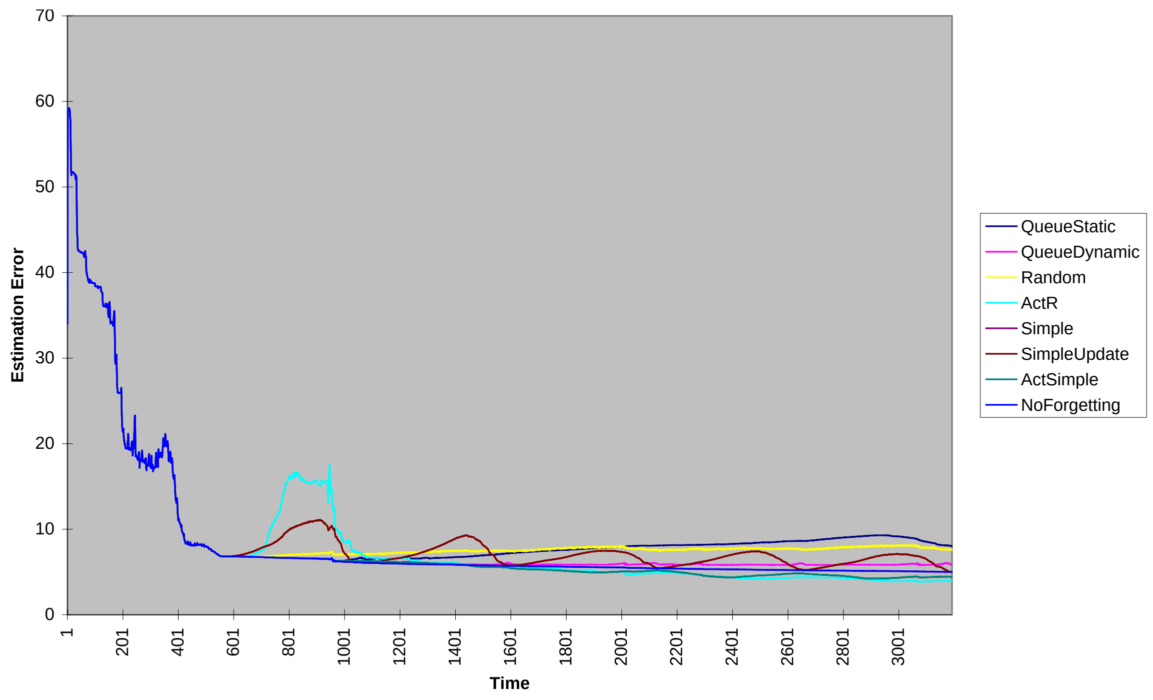


Figure A.6: Along path results for path 15 - 3 cycles 8 basestations

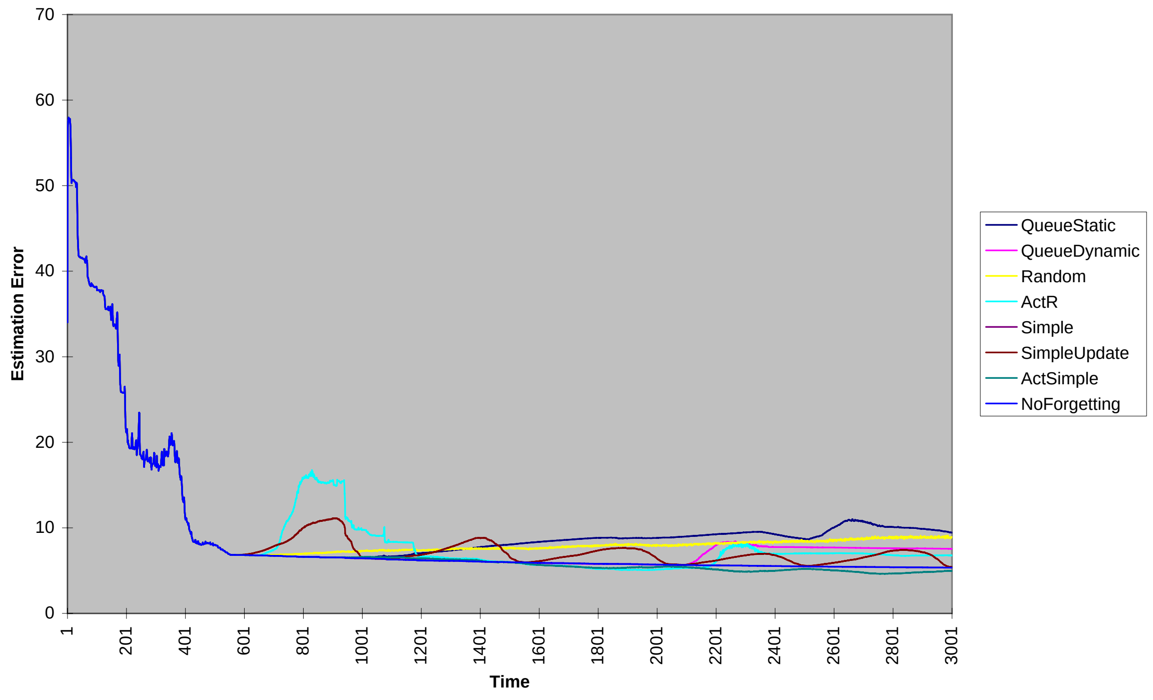


Figure A.7: Along path results for path 16 - 3 cycles 8 basestations

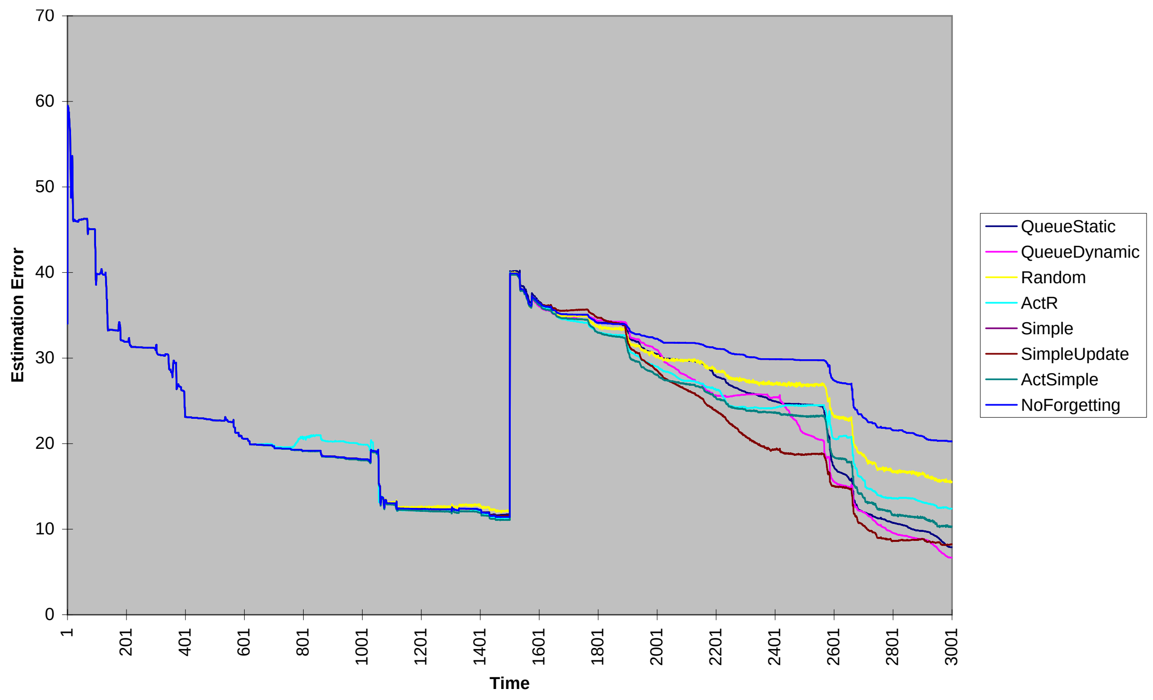


Figure A.8: Along path results for path 19 - random 8 - 4 basestations

A.2 Along the Path Error and Recall Maps

This section presents the complete set of error and recall maps for the Act-R, Queue Static, SIMPLE Update, and ActSimple algorithms. The maps for No Forgetting are also presented. Each set of maps is provided in video format and is located on the accompanying DVD in the path structure shown in Figure A.2.

- Chapter VIII Video Files
  - Act-R
  - ActSimple
  - No Forgetting
  - Queue Static
  - SIMPLE Update

Figure A.9: Directory Structure

Appendix B

Additional Components of Situational Awareness

SA is a complex phenomena that spans a diverse array of cognitive processes and properties. Many components of cognition can affect the ability to generate and maintain SA, although dependence on forgetting mechanisms is not universal. Some portions of the processes required to generate SA are not directly affected by data management issues and forgetting capabilities; however, these aspects of human cognition are still important. Benefits arising from components not directly associated with forgetting aid in the development of an environment suitable for the generation of SA, while the absence of these components can negate the benefits of forgetting. This appendix presents a number of cognitive phenomena that influence the ability to generate and maintain SA, while not directly being influenced by forgetting mechanisms.

B.1 System Stress

Human SA tends to suffer as human stress levels increase. In real world environments, many types of stressors exist, elements that affect information processing, but are not actually inherent components of the information to be processed. Stressors can be both environmental and physiological. Environmental stressors include noise, vibration, heat, cold, and poor lighting. Anxiety, fatigue, frustration, and anger are all considered to be physiological stressors. Four effects commonly arise from the presence of stressors. As stress builds, humans can undergo physiological changes, such as becoming frustrated. Short-term physiological effects also commonly occur, such as increased heart rate or a change in the output of catecholamines. In the presence of stressors, human information processing efficiency often decreases and long-term negative health effects may develop (Wickens et al., 2004).

The presence of stress can result in decreased perceptual abilities and inferior processing of resultant information. This degradation may cause a reduction in level one SA. As level one SA decreases, higher levels of SA start to degrade (Freedman and Adams, 2007).

B.2 Ideal, Achievable, and Actual Situational Awareness

Pew (2000) classifies human SA into three categories; ideal, achievable, and actual SA, as shown in Figure B.1. Ideal SA is often unachievable, as it represents an awareness of the entire situation, including items that may not be available under human cognitive and perceptual limitations. Achievable SA, a subset of ideal SA, captures the best level of SA for a given situation that is possible with human cognitive and perceptual capabilities. Actual SA represents a human's current level of SA, which is often a subset of the achievable SA

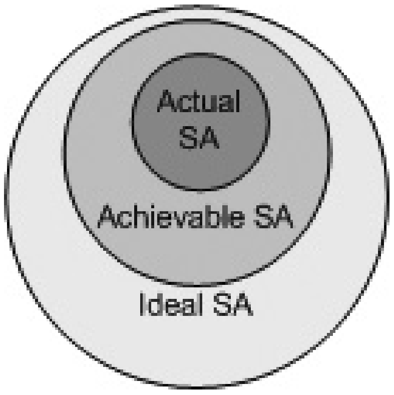


Figure B.1: Relationship between ideal, achievable, and actual SA (Recreated from (Pew, 2000))

category. Breaking down SA into ideal, achievable, and actual levels aids developers during the design and evaluation phases of interface development. Achievable SA provides performance goals for new interfaces and provides a valuable point of reference from which to compare a human's actual SA (Pew, 2000).

B.3 Vigilance

Maintaining good SA requires the constant observance of numerous specific details. In many domains, tasks can become quite boring and taxing. Humans have a tendency to stop paying attention to some of these details, at times to focus on other issues and sometimes to simply reduce the amount of required work. This phenomenon is known as loss of vigilance and can result in disastrous consequences. It is well known that human vigilance levels drop dramatically over time and have an effect on overall SA. Human vigilance levels often drop due to boredom and fatigue (Wickens and Hollands, 2000).

B.4 Uncertainty and Confidence

Uncertainty and confidence play a pivotal role in the relationship between human SA and performance. Human uncertainty often manifests itself in a human's hesitation or failure to act. Frequently, humans will continue to gather additional information in order to improve their awareness and confidence in actions selected to achieve desired objectives. However, complete certainty is almost always impossible. A trade off must constantly be made between the cost of acting under uncertainty and the benefits of additional search. Once an acceptable level of uncertainty has been reached, actions should be performed. When humans spend too much time searching the environment, the associated costs will increase and performance may subsequently degrade.

Many factors play into a person's level of confidence in their SA. When incoming data disagrees or outright contradicts itself, then uncertainty will increase as the likelihood that some previously acquired

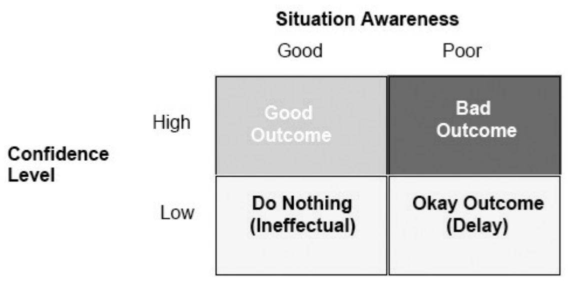


Figure B.2: SA and confidence levels (Endsley and Jones, 1997)

information is erroneous may be quite high. The quality and reliability of information sources within the environment can have an impact on confidence levels. Within adversarial domains, misleading or completely false information may also act to degrade confidence as the ability to trust incoming information is weakened. Additionally, the ability and confidence in that ability to process and correctly interpret incoming data can influence a person's faith in their SA.

As seen in Figure B.2, there are four possible states to define the relationship between confidence and SA, assuming a binary partitioning. The ideal case occurs when good SA is matched with high confidence, but care must be taken to handle the other scenarios. If high SA is achieved, but confidence is low, then performance may be unnecessarily poor, as observations will needlessly be sought from the environment. When SA is poor and uncertainty is high, then the desire to further monitor the environment will be appropriately high. The worst case occurs when SA is poor and confidence is high. Not only may poor action selection result while someone is in this state, but others may worsen their SA as they interact with the person acting with considerable confidence (Endsley and Jones, 1997).

B.5 Bad Situational Awareness vs. Bad Decision Making

SA is not synonymous with superior decision making, nor is it interchangeable with performance. Venturino et al. (1990) performed a study of fighter pilot SA where pilots participated in a combat simulation. Pilots who rated their SA as low performed poorly and pilots with self-assessments of average SA only performed at an average level of performance. When high self-ratings of SA were given, pilot performance varied greatly. From these results, Venturino et al. concluded that while SA may be necessary for high levels of performance, it is not the sole factor (Uhlarik and Comerford, 2002). Additionally, through luck or pure physical skill and ability, high levels of performance may occasionally be produced.

B.6 Biases and Expectations

Humans naturally have biases and expectations regarding different situations and their outcomes. In complex dynamic environments many objects require constant monitoring and surveillance. The patterns dictating how sensory channels are directed within the environment can be influenced by bottom-up factors (e.g., saliency and cues), but can also be affected by top-down biases and expectations. Previous experiences, goals, and mental models can all form expectations that govern the usage of sensory channels. Bias and expectations also influence higher levels of SA. In the presence of incomplete or ambiguous knowledge, humans often use expectations to filter choices and minimize cognitive load (Wickens and Hollands, 2000). Heuristics are often used by humans to complete tasks in time restricted situations. These heuristics can provide results comparable to more cognitively taxing mental processes, but are often domain and task specific (Gigerenzer et al., 1999). When inappropriately applied, heuristics can degrade performance and result in incorrect decision making (Wickens and Hollands, 2000).

B.6.1 Schemas and Mental Models

Mental models and schemas are constructs generally stored in long term memory that speed up comprehension of the environment and assist in determining the important of various SA constructs. Created through experience operating in some domain, mental models and schemas allow circumvention of various bottlenecks to SA by directing attention to the most importance aspects in the environment. Additionally, guidance on how to process incoming information and how to use it to project into the future can result from their use. Assistance in the decision making process is also possible by utilizing mental models and schemas (Endsley and Jones, 1997).

Considerable debate exists about the exact representation of humans' mental models. Minsky (1974, 1986) postulated that mental models are composed of frames. Frames are considered to be the basic building block of knowledge with a human's memory consisting of millions of these frames in a hierarchical data structure. Lower level frames will possess greater specificity of a particular task or event, but at a smaller scope. Within each frame, top level components are fixed, in that they represent items mandatory to the domain or phenomena being represented. As a progression is made to lower items within a frame, items gain in optionality. Many of the items within a frame may already possess values even before the frame is applied to an instance of a situation. These values act as defaults, assisting in the understanding of complex situations and causing deviations from the default values to be noticed. Defaults can negatively impact recall in some situations. If the real value of some feature does not agree with a default, but this fact is not noticed and recorded, then the default value can erroneously be stored for recall later. Minsky defined six different types of frames, shown in Table B.1. While all of the frame types possess the same basic characteristics, hierarchy

Table B.1: Minsky's variety of frames (Minsky, 1974)

Frames for objects
Temporal or programmatic frames
Mixed frames for situations
Grammar frames
Narrative or text frames
Scientific paradigms

and leveling of optionality, each is used for a different purpose.

When the environment highly correlates with a mental model, the decision making process is even further speed up, as less processing is required. If a mental model partially disagrees with the current state of the environment, benefits can still be realized from its use due to human pattern matching abilities and utilization of best fit approximation. Additionally, mental models provide default values for missing information, allowing for decision making and prediction even in the presence of uncertainty. Confidence in the model can also negatively affect this process as doubt about a mental model's correctness and applicability increases, reliance on the model will decrease until it is not used at all (Endsley and Jones, 1997).

B.6.2 Framing and Order

The order in which observations about the environment are made can have a substantial impact on a human's ability to understand a situation. This impact may result from the formation of a situational model about the environment and the current task at hand, which is primarily composed of these observations. As new information is received, the model is altered, ideally resulting in a more detailed and accurate picture of the state of the environment. Much of this model is necessarily built from inferences and assumptions, since complex dynamic environments contain too much information and uncertainty to allow for full understanding. When incorrect ordering of information is presented to a human, they may either produce incorrect inferences or fail to produce them altogether (Sanford and Garrod, 1981).

Dooling and Lachman (1971) crafted a demonstration of this phenomenon with a short text passage.

> "With hocked gems financing him, our hero bravely defied all scornful laughter that tried to prevent his scheme. Your eyes deceive', he had said. An egg, not a table, correctly typifies this unexplored planet'. Now three sturdy sisters sought proof. Forging along, sometimes through calm vastness, yet more often turbulent peaks and valleys, days became weeks as many doubters spread fearful rumors about the edge. At last, from nowhere, welcome winged creatures appeared signifying momentous success."

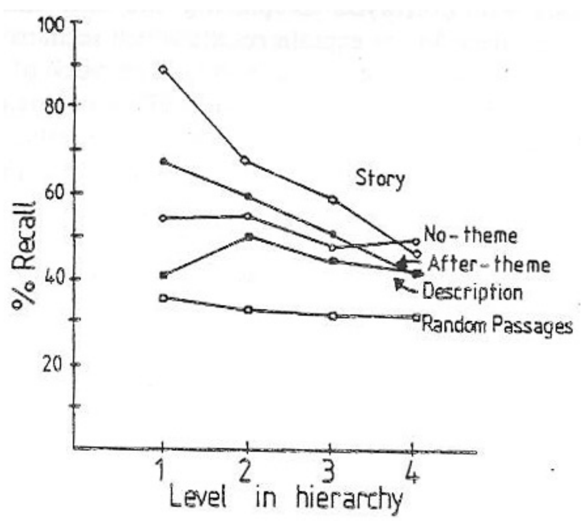


Figure B.3: Recall performance of details from a reading passage ((Sanford and Garrod, 1981) recreated from (Thorndyke, 1977)

This passage is generally considered difficult to remember, much less interpret and understand, until the title is provided, “Christopher Columbus’s discovery of America”. Once the title is available to frame the text, a basic understanding can start to form, but not until the passage is reread can many of the rich metaphors be appreciated (Sanford and Garrod, 1981).

Thorndyke (1977) experimentally tested the effects of information ordering by presenting a reading passage to subjects whose recall was subsequently tested. Five different versions of the passage were used during the test, Story (the passage was presented in its normal format and ordering), After-Theme (sentences containing the theme of the story were moved to the end of the passage after some slight restructuring), No-Theme (the passage had its theme completely removed), Description (the passage was altered so that only nominal and pronominal references remained), and Random Passages. Results from this experiment can be seen in Figure B.3. In the graph, Level of Hierarchy refers to the relative importance of a piece of information to the overall meaning and context of the passage. From this experiment, Thorndyke was able to conclude that the standard story format ordering of a passage allowed readers to form a global frame (in the Minsky sense) of reference that subsequent information could refine. The After-Theme and No-Theme versions prevented the reader from constructing a global frame, but allowed the readers to utilize local causal’ links and simple temporal ordering (Sanford and Garrod, 1981).

Table B.2: Negative forces acting on SA (Endsley and Jones, 1997)

Error in observations
Denial of observations
False observations
Overloading of observations
Inducing wrong conclusions
Disorganization of observations
Induce incorrect assumptions
Increase of unpredictability

B.6.3 Observation Quality and Availability

The quality of SA that can be obtained is based solely on the quality of the input observations from the environment and the ability of a human's cognitive processes to generate an understanding of the environment and future actions. While producing a highly efficient and effective SA capability is difficult with even the best set of observations in the real world, false and misleading information will result in a direct assault on this capability. Causes of these undesired observations do not necessarily need to originate from adversarial forces in the environment, such as direct competitors, but may come from teammates or even neutral agents who happen to be present in the environment. In this case, agent can refer to practically anything, humans, animals, or even machines that are somehow involved with the current process. Additionally, a human's own senses may help to undermine the ability to generate high SA because of failures and errors in the ability to perceive the environment (Endsley and Jones, 1997).

Table B.2 partitions the collection of negative forces acting on SA into eight separate areas. The first, and most obvious, is simply erroneous observations of the environment. Sources of erroneous observations are numerous, ranging from sensory error to the inability of human senses to detect characteristics about objects in the environment. Second in the list, denial of observations, may result when active entities in the environment are actively attempting to prevent particular observations from being made. Third, false observations help to increase uncertainty, generate false beliefs, and decrease trust in valid percepts. Other negative forces acting against SA can impose similar constraints and degrade the ability for a human to achieve and maintain high levels of SA (Endsley and Jones, 1997).

B.7 Workload

Human SA is affected by both cognitive and physical workload, which can act as a stressor when present in excess (Wickens et al., 2004). Figure B.4 depicts the relationship between workload and SA. Under low to moderate workload, SA and workload can vary almost independently. At the lowest levels of workload, vigilance may decrease and attention can be diverted from important task elements. As workload exceeds

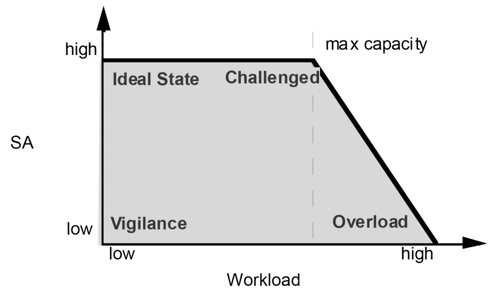


Figure B.4: SA vs. Workload (Endsley, 1993)

available capacity, performance and SA decrease. Under this condition, every task requiring attention cannot be adequately handled and information regarding the environment is ignored or undetected (Endsley and Jones, 1997). Under these high workload conditions, critical information and tasks are often neglected, forgotten, or ignored (Wickens et al., 2004). Missing information directly degrades level one SA and indirectly affects higher levels of SA. Level two and level three SA are also directly affected by the time pressures resulting from an increased workload. Without adequate cognitive capacity, level one constructs cannot be formed into level two constructs and the development of level three constructs equally suffers. Forgetting stale, irrelevant, and out of date data may reduce cognitive workload by reducing the mental search time required to recall items necessary for successful task completion.

B.8 Data Driven and Goal Driven Behavior

Two diametrically opposed processes can be used for operating in a complex dynamic environment, data driven processing and goal driven processing. When operating in a data driven mindset, the environment is scanned in parallel for salient signal properties to emerge. Salient features of the environment are observed and processed into level one SA. Patterns can easily be detected when utilizing a data driven approach, which can be used to modify or replace any current goals. Goal driven processing involves directed sampling in the environment. Using this approach, specific information from the environment can be sought out and then evaluated with respect to current goals, forming level two SA.

Choosing which mode to use can play a significant role in the success of a task, as the selection will affect how attention is directed and how information is perceived and interpreted. These two methods need to be alternated constantly, since data driven processing can lead to data overflow and goal driven processing can result in too narrow of a focus and the loss of the ability to determine information’s importance (Endsley and Jones, 1997).

BIBLIOGRAPHY

Adams, J. A. (2007). Unmanned vehicle situation awareness: A path forward. In *Proceedings of the 2007 Human Systems Integration Symposium*.

Adams, J. A. (2009). Multiple robot-single human interaction: Effects on perceived workload. *Behavior & Information Technology*, 28(2):183–198.

Adams, J. A. and Freedman, S. T. (2007). Unmanned system autonomy, situation awareness, and system safety. In *Proceedings of the 25th International System Safety Conference*, pages 800–810.

Adams, J. A., Humphrey, C. M., Goodrich, M. A., Cooper, J. L., Morse, B. S., Engh, C., and Rasmussen., N. (2009). Cognitive task analysis for developing unmanned aerial vehicle wilderness search support. *Journal of Cognitive Engineering and Decision Making*, 3(1):1–26.

Adams, M. J., Tenney, Y. J., and Pew, R. W. (1995). Situation awareness and the cognitive management of complex systems. 37(1):85–104.

Allport, A. and Wylie, G. (2000). Task-switching, stimulus-response binding, and negative priming. In Monsell, S. and Driver, J. S., editors, *Control of Cognitive Processes: Attention and Performance XVIII*, pages 35–70. MIT Press, Cambridge, MA.

Altmann, E. M. (2002). Functional decay of memory for tasks. *Psychological Research*, 66:287–297.

Altmann, E. M. and Gray, W. D. (2002). Forgetting to remember: The functional relationship of decay and interference. *Psychological Science*, 13(1):27–33.

Altmann, E. M. and Schunn, C. D. (2002). Integrating decay and interference: A new look at an old interaction. In *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, pages 65–70. Lawrence Erlbaum Associates, Mahwah, NJ.

Ambrose, R. O., Aldridge, H., Askew, R. S., Burridge, R. R., Bluethmann, W., Diftler, M., Lovchik, C., Magruder, D., and Rehnmark, F. (2000). Robonaut: NASA's space humanoid. *IEEE Intelligent Systems*, 15(4):57–63.

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

Anderson, J. R. and Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Mahwah, NJ.

Anderson, J. R. and Matessa, M. P. (1997). A production system theory of serial memory. *Psychological Review*, 104:728–748.

Anderson, J. R. and Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science*, 2:396–408.

Anderson, R. B., Doherty, M. E., Berg, N. D., and Friedrich, J. C. (2005). Sample size and the detection of correlation - a signal detection account: Comment on kareev (2000) and juslin and olsson (2005). *Psychological Review*, 112(1):268–279.

Arkin, R. C. (1998). *Behavior-based Robotics*. MIT Press, Cambridge, MA.

Atkinson, R. C. and Shiffrin, R. M. (1971). The control of short-term memory. *Scientific American*, 225(2):82–90.

Australian Civil Aviation Authority (1994). *Air Traffic Control Abinitio Training Program*. Australian Civil Aviation Authority, Air Traffic Services, ATS Human Resources, Canberra, Australia.

Baddeley, A. D. (1986). *Working Memory*. Oxford University Press, New York, NY.

Baddeley, A. D. (2000). The episodic buffer: A new component of working memory? *Trends in Cognitive Sciences*, 4(11):417–423.

Balch, T. (2002). Taxonomies of multirobot task and reward. In Balch, T. and Parker, L. E., editors, *Robot Teams*, pages 23–35. A. K. Peters, Natick, MA.

Basu, P. and Redi, J. (2004). Movement control algorithms for realization of fault-tolerant ad hoc robot networks. *IEEE Network*, 18(4):36–44.

Breazeal, C., Gray, J., and Berlin, M. (2009). An embodied cognition approach to mindreading skills for socially intelligent robots. *The International Journal of Robotics Research*, 28(5):656–680.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23.

Brooks, R. A. (1991). Intelligence without reason. In *Proceedings of the 12th International Joint Conference of Artificial Intelligence*, pages 569–595.

Brown, G. D. A., Chater, N., and Neath, I. (2007). A temporal ratio model of memory. *Psychological Review*, 114(3):539–576.

Brown, G. D. A. and Hulme, C. (1995). Modeling item length effects in memory span: No rehearsal needed? *Journal of Memory and Language*, 34:594–621.

Brown, G. D. A., Preece, T., and Hulme, C. (2000). Oscillator-based memory for serial order. *Psychological Review*, 107:127–181.

Burgess, N. and Hitch, G. (1992). Toward a network model of the articulatory loop. *Journal of Memory and Language*, 31:429–460.

Burgess, N. and Hitch, G. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review*, 106:551–581.

Burghart, C., Gaertner, C., and Woern, H. (2006). Cooperative solving of a children's jigsaw puzzle between human and robot: First results. In Beetz, M., Rajan, K., Thielscher, M., and Rusu, R. B., editors, *Cognitive Robotics: Papers from the AAAI Workshop*, number WS-06-03, pages 33–39.

Burke, J. L. and Murphy, R. R. (2004). Situation awareness and task performance in robot-assisted technical search: Bujold goes to Bridgeport. Technical Report CRASAR-TR2004-23, Center for Robot Assisted Search & Rescue, University of South Florida.

Byrne, M. D. (1998). Taking a computational approach to aging: The SPAN theory of working memory. *Psychology and Aging*, 13:309–322.

Callan, D. E. and Schweighofer, N. (2010). Neural correlates of the spacing effect in explicit verbal semantic encoding support the deficient-processing theory. *Human Brain Mapping*, 31:645–659.

Cassimatis, N. (2006). A cognitive substrate for achieving human-level intelligence. *AI Magazine*, 27(2):45–56.

Cassimatis, N., Bugajska, M., Dugas, S., Murugesan, A., and Bello, P. (2007). An architecture for adaptive algorithmic hybrids. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 1520–1526.

Cassimatis, N., Mueller, E. T., and Winston, P. H. (2006). Achieving human-level intelligence through integrated systems and research. *AI Magazine*, 27(2):12–14.

Cassimatis, N. L. (2002). *Polyscheme: A Cognitive Architecture for Integrating Multiple Representation and Inference Schemes*. PhD thesis, Massachusetts Institute of Technology.

Cepeda, N. J., Pashler, H., Vul, E., Wixted, J., and Rohrer, D. (2006). Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132:354–380.

Choi, D., Kaufman, M., Langley, P., Nejati, N., and Shapiro, D. (2004). An architecture for persistent reactive behavior. In *Proceedings of the 3rd International Conference on Joint Autonomous Agents and Multiagent Systems*, pages 988– 995.

Chong, R. S. (2003). The addition of an activation and decay mechanism to the Soar architecture. In *The Logic of Cognitive Systems: Proceedings of the 5th International Conference on Cognitive Modeling*, pages 45–50.

Correia, L. and Abreu, A. (2004). Forgetting and fatigue in mobile robot navigation. In *Advances in Artificial Intelligence: Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, pages 434–443. Springer, Sao Luis, Brazil.

Cox, M. T. (2005). Metacognition in computation: A selected research review. *Artificial Intelligence*, 169(2):104–141.

Cox, M. T. (2007). Perpetual self-aware cognitive agents. *AI Magazine*, 28(1):32–45.

Cuddy, L. J. and Jacoby, L. L. (1982). When forgetting helps memory: An analysis of repetition effects. *Journal of Verbal Learning and Verbal Behavior*, 21:451–467.

Daily, L. Z., Lovett, M. C., and Reder, L. M. (2001). Modeling individual differences in working memory performance: A source activation account. *Cognitive Science*, 25(3):315–353.

DAKOTA (2010). Design analysis kit for optimization and terascale applications. http://www.cs.sandia.gov/dakota.

Davids, K., Smith, L., and Martin, R. (1991). Controlling system uncertainty in sport and work. *Applied Ergonomics*, 22(2):312–315.

Davies, E. R. (2004). *Machine Vision: Theory, Algorithms, Practicalities*. Morgan Kaufmann, San Francisco, CA.

Dell (2003). Dell TrueMobile 1150 Mini PCI Card. http://support.dell.com/support/edocs/network/tm1150mp/en/index.htm.

Dempster, F. N. (1988). The spacing effect: A case study in the failure to apply the results of psychological research. *American Psychologist*, 43(8):627–634.

Dominguez, C. (1994). Can SA be defined? Technical Report AL/CF-TR-1994-0085, Air Force Systems Command, Wright-Patterson Air Force Base. pages 5-15.

Dooling, D. J. and Lachman, R. (1971). Effects of comprehension on retention of prose. *Journal of Experimental Psychology*, 88:216–222.

Dosher, B. (1999). Item interference and time delays in working memory: Immediate serial recall. *International Journal of Psychology, Special Issue: Short-Term/Working Memory*, 34:276–284.

Dosher, B. and Ma, J. (1998). Output loss or rehearsal loop? Output-time versus pronunciation-time limits in immediate recall for forgetting-matched materials. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 24(2):316–335.

Drury, J. L., Scholtz, J., and Yanco, H. A. (2003). Awareness in human-robot interactions. In *Proceedings of the 2003 IEEE Conference on Systems, Man, and Cybernetics*, pages 912– 918.

Dudek, G., Jenkins, M., and Milios, E. (2002). A taxonomy of multirobot systems. In Balch, T. and Parker, L. E., editors, *Robot Teams*, pages 3–22. A. K. Peters, Natick, MA.

Dudziak, W. (2007). Presentation and analysis of a multi-dimensional interpolation function for non-uniform data: Microsphere projection. Master's thesis, University of Akron.

Ebbinghaus, H. (1885). *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, New York, NY. translators Ruger, H. A. and Bussenius, C. E. 1913.

Elliott, S. W. and Anderson, J. R. (1995). Effect of memory decay on predictions from changing categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 21(4):815–836.

Endsley, M. R. (1988a). Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors Society 32nd Annual Meeting*, pages 97–101.

Endsley, M. R. (1988b). Situation awareness global assessment technique (SAGAT). In *Proceedings of the National Aerospace and Electronics Conference*, pages 789–795.

Endsley, M. R. (1993). Situation awareness and workload: Flip sides of the same coin. In Jensen, R. S. and Neumeister, D., editors, *Proceedings of the 7th International Symposium on Aviation Psychology*, pages 906–911.

Endsley, M. R. (1995a). Measurement of situation awareness in dynamic systems. *Human Factors*, 37(1):65–84.

Endsley, M. R. (1995b). Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64.

Endsley, M. R. and Jones, W. M. (1997). Situation awareness information dominance & information warfare. Technical Report AL/CF-TR-1997-0156, United States Air Force Armstrong Laboratory.

Estes, W. K. (1955). Statistical theory of distributional phenomena in learning. *Psychological Review*, 62(5):369–377.

Farrell, S. and Lewandowsky, S. (2002). An endogenous distributed model of ordering in serial recall. *Psychonomic Bulletin and Review*, 9:59–79.

Flach, J. M. (1995). Situation awareness: Proceed with caution. *Human Factors*, 37(1):149–157.

Fox, D., Burgard, W., and Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Fracker, M. L. (1988). A theory of situation assessment: Implications for measuring situation awareness. In *Proceedings of the Human Factors Society 32nd Annual Meeting*, pages 102–106.

Freedman, S. T. and Adams, J. A. (2007). The inherent components of unmanned vehicle situation awareness. In *Proceedings of the 2007 IEEE International Conference on Systems, Man, and Cybernetics*, pages 973–977.

Freedman, S. T. and Adams, J. A. (2008). Synthetic cognitive agent situational awareness components. In *AAAI Technical Report FS-08-04*, page 62.

Gaissmaier, W., Schooler, L. J., and Rieskamp, J. (2006). Simple predictions fueled by capacity limitations: When are they successful? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 32(5):966–982.

Gatsoulis, I. (2008). *Performance Metrics and Human-Robot Interaction for Teleoperated Systems*. PhD thesis, School of Mechanical Engineering, University of Leeds.

Gatsoulis, Y. and Virk, G. S. (2006). Modular situational awareness for CLAWAR robots. In Tokhi, M. O., Virk, G. S., and Hossain, M. A., editors, *Climbing and Walking Robots: Proceedings of the 8th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 1011–1020. Springer, Berlin, Germany; Heidelberg, Germany; New York, NY.

Gerkey, B. and Matarić, M. J. (2003). A framework for studying multi-robot task allocation. In Schultz, A. C., Parker, L. E., and Schneider, F. E., editors, *Multi-Robot Systems: From Swarms to Intelligent Automata - Proceedings of the 2003 International Workshop on Multi-Robot Systems*, volume 2, pages 15–26. Springer.

Gigerenzer, G. and Brighton, H. (2009). Homo heuristicus: Why biased minds make better inferences. *Topics in Cognitive Science*, 1(1):107–143.

Gigerenzer, G., Todd, P. M., and the ABC Research Group (1999). *Simple Heuristics That Make Us Smart*. Oxford University Press, New York.

Giralt, G., Sobek, R., and Chatila, R. (1979). A multi-level planning and navigation system for a mobile robot: A first approach to Hilare. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pages 335–337.

Glenberg, A. M. (1979). Component-levels theory of the effects of spacing of repetitions on recall and recognition. *Memory and Cognition*, 7(2):95–112.

Glenn, F., Schwartz, S., and Ross, L. (1992). Development of a human operator simulator version V (HOS-V): Design and implementation. Research Note 92-PERI-POX. Army Research Institute.

Goldstein, D. G. and Gigerenzer, G. (2002). Models of ecological rationality: The recognition heuristic. *Psychological Review*, 109(1):75–90.

Goodrich, M. A., Morse, B. S., Gerhardt, D., Cooper, J. L., Quigley, M., Adams, J. A., and Humphrey, C. M. (2008). Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics*, 25(1-2):89–110.

Goodrich, M. A. and Schultz, A. C. (2007). Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275.

Hanford, S. D., Janrathitikarn, O., and Long, L. N. (2008). Control of a six-legged mobile robot using the Soar cognitive architecture. In *Proceedings of the 46th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA Paper No. 2008-0878.

Harper, K. A. and Zacharias, G. L. (2004). Common problems and helpful hints to solve them: Lessons learned in integrating cognitive models in large-scale simulation environments. In Ingalls, R. G., Rossetti, M. D., Smith, J. S., and Peters, B. A., editors, *Proceedings of the 2004 Winter Simulation Conference*, pages 891–897.

Harrison, A. M. (2007). *Online of Offline? Exploring Working Memory Constraints in Spatial Updating*. PhD thesis, University of Pittsburgh.

Harrison, T. and Schunn, C. (2003). Segmented spaces: Coordinated perception of space in ACT-R. In Detje, F., Dorner, D., and Schaub, H., editors, *The Logic of Cognitive Systems: Proceedings of the 5th International Conference on Cognitive Modeling*, page 307. Universitäts-Verlag Bamberg, Bamberg, Germany.

Hawes, N., Brenner, M., and Sjöö, K. (2009). Planning as an architectural control mechanism. In *Proceedings of the International Conference on Human-Robot Interaction*, pages 229–230, La Jolla, CA.

Hawes, N., Hanheide, M., Sjöö, K., Aydemir, A., Jensfelt, P., Göbelbecker, M., Brenner, M., Zender, H., Lison, P., Kruijff-Korbayová, I., Kruijff, G.-J., and Zillich, M. (2010). Dora the explorer: A motivated robot. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1617–1618, Toronto, Canada.

Hawes, N., Sloman, A., Wyatt, J., Zillich, M., Jacobsson, H., Kruijff, G.-J. M., Brenner, M., Berginc, G., and Skocaj, D. (2007). Towards an integrated robot with multiple cognitive functions. In *Proceedings of the 22nd Conference on Artificial Intelligence*, pages 1548–53.

Hawes, N., Wyatt, J., and Sloman, A. (2006). An architecture for embodied cognitive systems. Technical Report CSR-06-12, University of Birmingham.

Henson, R. N. A. (1998). Short-term memory for serial order: The start-end model. *Cognitive Psychology*, 36:73–137.

Hoffman, G. and Breazeal, C. (2007). Effects of anticipatory action on human-robot teamwork: Efficiency, fluency, and perception of team. In *Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interaction*, pages 1–8.

Howell, J. and Donald, B. R. (2000). Practical mobile robot self-localization. In *Proceedings of the 2000 International Conference on Robotics and Automation*, pages 3485–3492.

Hulme, C., Roodenrys, S., Schweickert, R., Brown, G. D. A., Martin, S., and Stuart, G. (1997). Word-frequency effects on short-term memory tasks: Evidence for a redintegration process in immediate serial recall. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 23:1217–1232.

Humphrey, C. M. and Adams, J. A. (2008). Compass visualizations for human-robot interaction. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*, pages 239–246, New York, NY.

Humphrey, C. M. and Adams, J. A. (2009a). General visualization abstraction algorithm for directable interfaces: Component performance and learning effects. *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, Accepted.

Humphrey, C. M. and Adams, J. A. (2009b). Robotic tasks for CBRNE incident response. *Journal of Advanced Robotics, Special Issue on Disaster Response Robotics*, 23:1217–1232.

International Commission on Illumination (1932). *Commission internationale de l'Eclairage proceedings*. Cambridge University Press, Cambridge, England.

Jacobsson, H., Hawes, N., Kruijff, G.-J., and Wyatt, J. (2008). Crossmodal content binding in information-processing architectures. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*.

Johnson, T. R. (1997). Control in ACT-R and Soar. In Shafto, M. and Langley, P., editors, *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, pages 343–348. Lawrence Erlbaum Associates, Hillsdale, NJ.

Jonides, J., Lewis, R. L., Nee, D. E., Lustig, C. A., Berman, M. G., and Moore, K. S. (2008). The mind and brain of short-term memory. *Annual Review of Psychology*, 59:193–224.

Kaber, D. B., Onal, E., and Endsley, M. R. (2000). Design of automation for telerobots and the effect on performance, operator situation awareness, and subjective workload. *Human Factors and Ergonomics in Manufacturing*, 10(4):409–430.

Kareev, Y. (2005). And yet the small-sample effect does hold: Reply to juslin and olsson (2005) and anderson, doherty, berg, and friedrich (2005). *Psychological Review*, 112(1):280–285.

Kieras, D. E., Meyer, D. E., Mueller, S., and Seymour, T. (1999). Insights into working memory from the perspective of the EPIC architecture for modeling skilled perceptual-motor and cognitive human performance. In Miyake, A. and Shah, P., editors, *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control*, pages 183–223. Cambridge University Press, New York.

Kira, Z. and Arkin, R. C. (2004). Forgetting bad behavior: Memory management for case-based navigation. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robotics and Systems*, volume 4, pages 3145–3152.

Kruijff, G.-J. M., Lison, P., Benjamin, T., Jacobsson, H., and Hawes, N. (2007). Incremental, multi-level processing for comprehending situated dialogue in human-robot interaction. In Lopes, L. S., Belpaeme, T., and Cowley, S. J., editors, *Proceedings of the 2007 Symposium on Language and Robots*, pages 55–64.

Kuhl, B. A., Dudukovic, N. M., Kahn, I., and Wagner, A. D. (2007). Decreased demands on cognitive control reveal the neural processing benefits of forgetting. *Nature Neuroscience*, 10(7):908–914.

Kumar, V. K. (1971). The structure of human memory and some educational implications. *Review of Educational Research*, 41(5):379–417.

Laird, J. E. (2001). It knows what you're going to do: Adding anticipation to a quakebot. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 385–392.

Laird, J. E. (2006). The Soar 8 tutorial: Part 3. Online. http://ai.eecs.umich.edu/soar/sitemaker/docs/tutorial/TutorialPart3.pdf.

Laird, J. E. (2008). Extending the Soar cognitive architecture. In *Proceedings of the 1st Conference on Artificial General Intelligence*, pages 224–235.

Landauer, T. K. (1969). Reinforcement as consolidation. *Psychological Review*, 76(1):82–96.

Lane, N., Strieb, M., Glenn, F., and Wherry, R. (1981). The human operator simulator: An overview. In Moraal, J. and Kraiss, K.-F., editors, *Manned Systems Design: Methods, Equipment, and Applications*. Plenum Press, New York, NY.

Langley, P. and Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1469–1474.

Langley, P., Laird, J. E., and Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160.

Lee, C. L. and Estes, W. K. (1977). Order and position in primary memory for letter strings. *Journal of Verbal Learning and Verbal Behavior*, 16:395–418.

Lee, C. L. and Estes, W. K. (1981). Item and order information in short-term memory: Evidence for multilevel perturbation processes. *Journal of Experimental Psychology: Human Learning and Memory*, 7:149–169.

Lehman, J. F., Laird, J., and Rosenbloom, P. (2006). A gentle introduction to Soar, an architecture for human cognition: 2006 update. Technical report, University of Michigan.

Lewandowsky, S., Duncan, M., and Brown, G. D. A. (2004). Time does not cause forgetting in short-term serial recall. *Psychonomic Bulletin & Review*, 11(5):771–790.

Lewandowsky, S. and Murdock, B. B. (1989). Memory for serial order. *Psychological Review*, 96:25–57.

Lewis, R. L. (1993). An architecturally-based theory of sentence comprehension. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, pages 108–113. Lawrence Erlbaum, Hillsdale, NJ.

Likhachev, M. and Arkin, R. C. (2001). Spatio-temporal case-based reasoning for behavioral selection. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 1627–1634.

Likhachev, M., Kaess, M., and Arkin, R. C. (2002). Learning behavioral parameterization using spatio-temporal case-based reasoning. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1282–1289.

Lison, P., Ehrler, C., and Kruijff, G.-J. M. (2010). Belief modeling for situation awareness in human-robot interaction. Technical report, Language Technology Lab, German Research Centre for Artificial Intelligence, Saarbrücken, Germany.

Luria, A. R. (1968). *The Mind of a Mnemonist: A Little Book about a Vast Memory*. Basic Books Inc., New York, NY. translator Lynn Solotaroff.

Lustig, C., Matell, M. S., and Meck, W. H. (2005). Not "just" a coincidence: Frontal-striatal interactions in working memory and interval timing. *Memory*, 13(3/4):441–448.

MacKenzie, D. C., Arkin, R. C., and Cameron, J. M. (1997). Multiagent mission specification and execution. *Autonomous Robots*, 4(1):29–52.

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., and Stokes, D. (2004). AI characters and directors for interactive computer games. In *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference*.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Markovitch, S. and Scott, P. D. (1988). The role of forgetting in learning. In *Proceedings of the 5th International Conference on Machine Learning*, pages 459–465. Morgan Kaufmann, San Mateo, CA.

Markovitch, S. and Scott, P. D. (1989). Information filters and their implementation in the SYLLOG system. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 404–407. Morgan Kaufmann.

Matarić, M. J. (1992). Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–311.

Matarić, M. J. (2007). *The Robotics Primer*. The MIT Press, Cambridge, MA.

McCarthy, J. (1959). Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91. Her Majesty's Stationary Office, London, UK.

McCarthy, J., Minsky, M., Sloman, A., Gong, L., Lau, T., Morgenstern, L., Mueller, E. T., Riecken, D., Singh, M., and Singh, P. (2002). An architecture of diversity for commonsense reasoning. *IBM Systems Journal*, 41(3):530–539.

Melton, A. W. (1967). Repetition and retrieval from memory. *Science*, 158(3800):532.

Miller, C. S. and Laird, J. E. (1996). Accounting for graded performance within a discrete search framework. *Cognitive Science*, 20:499–537.

Minsky, M. (1974). A framework for representing knowledge. Technical Report AIM-306, Massachusetts Institute of Technology.

Minsky, M. (1986). *The Society of Mind*. Simon & Schuster, New York, NY.

Minsky, M. (2006). *The Emotion Machine*. Simon & Schuster, New York, NY.

Minsky, M., Singh, P., and Sloman, A. (2004). The St. Thomas common sense symposium: Designing architectures for human-level intelligence. *AI Magazine*, 25(2):113–124.

Minsky, M. L. (1992). Future of AI technology. *Toshiba Review*, 47(7).

Mooney, R. (1989). The effect of rule use on the utility of explanation-based learning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 725–730, San Francisco, CA. Morgan Kaufmann.

Moravec, H. (1990). The Stanford cart and the CMU rover. In Cox, I. J. and Wilfong, G. T., editors, *Autonomous Robot Vehicles*, pages 407–441. Springer-Verlag, New York, NY.

Mower, E. K., Feil-Seifer, D. J., Matarić, M. J., and Narayanan, S. (2007). Investigating implicit cues for user state estimation in human-robot interaction using physiological measurements. In *Proceedings of the 2007 IEEE International Workshop on Robot and Human Interactive Communication*, pages 1125–1130.

Mueller, E. T. (2006). *Commonsense Reasoning*. Morgan Kaufman.

Mueller, S. T. (2002). *The Roles of Cognitive Architecture and Recall Strategies in Performance of the Immediate Serial Recall Task.* PhD thesis, University of Michigan.

Mueller, S. T. and Krawitz, A. (2009). Reconsidering the two-second decay hypothesis in verbal working memory. *Journal of Mathematical Psychology*, 53(1):14–25.

Murdock, B. B. (1982). A theory for the storage and retrieval of item and associative information. *Psychological Review*, 89(6):609–626.

Murphy, R. R., Casper, J., Micire, M., and Hyams, J. (2000). Mixed-initiative control of multiple heterogeneous robots for urban search and rescue. Technical Report CRASAR-TR2000-11, Center for Robot Assisted Search & Rescue, University of Southern Florida.

Murphy, R. R., Pratt, K. S., and Burke, J. L. (2008). Crew roles and operational protocols for rotary-wing micro-UAVs in close urban environments. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction*, pages 73–80.

Mutlu, B., Shiwa, T., Kanda, T., Ishiguro, H., and Hagita, N. (2009). Footing in human-robot conversations: How robots might shape participant roles using gaze cues. In *Proceedings of the 4th ACM/IEEE International Conference on Human-Robot Interaction*, pages 61–68.

Nairne, J. S. (2002). Remembering over the short-term: The case against the standard model. *Annual Review of Psychology*, 53:53–81.

Naveh, I. and Sun, R. (2006). Simulating a simple case of organizational decision making. In *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, pages 124–150. Cambridge University Press, New York, NY.

Neath, I. and Nairne, J. S. (1995). Word-length effects in immediate memory: Overwriting trace decay theory. *Psychonomic Bulletin and Review*, 2:429–441.

Neisser, U. (1976). *Cognition and Reality: Principles and Implications of Cognitive Psychology*. W. H. Freeman & Company, San Francisco, CA.

NetStumbler (2002). NetStumbler 0.4.0. http://www.stumbler.net/.

New, J., Cosmides, L., and Tooby, J. (2007). Category-specific attention for animals reflects ancestral priorities, not expertise. *Proceedings of the National Academy of Sceinces of the United States of America*, 104(42):16598–16603.

New York Times News Service (2000). Study finds trends in incompetence. The Tennessean. Nashville, TN.

Nuxoll, A., Laird, J. E., and James, M. R. (2004). Comprehensive working memory activation in Soar. In *Proceedings of the 6th International Conference on Cognitive Modeling*, pages 226–230.

Nuxoll, A. M. and Laird, J. E. (2007). Extending cognitive architecture with episodic memory. In *Proceedings of the 22nd Conference on Artificial Intelligence*, pages 1560–1565.

Oberauer, K. (2006). Is the focus of attention in working memory expanded through practice? *Journal of Experimental Psychology*, 32(2):197–214.

Okuno, Y., Kanda, T., Imai, M., Ishiguro, H., and Hagita, N. (2009). Providing route directions: Design of robot's utterance, gesture, timing. In *Proceedings of the 4th ACM/IEEE International Conference on Human-Robot Interaction*, pages 53–60.

Page, M. P. A. and Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review*, 105(4):761–781.

Parasuraman, R., Sheridan, T. B., and Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 30(3):286–297.

Pavlik, P. I. and Anderson, J. R. (2003). An ACT-R model of the spacing effect. In Detje, F., Dörner, D., and Schaub, H., editors, *Proceedings of the 5th International Conference on Cognitive Modeling*, pages 177–182. Universitäts-Verlag Bamberg, Bamberg, Germany.

Pavlik Jr., P. I. and Anderson, J. R. (2005). Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science*, 29:559–586.

Pew, R. W. (2000). The state of situation awareness measurement: Heading toward the next century. In Endsley, M. R. and Garland, D., editors, *Situation Awareness Analysis and Measurement*, pages 33–47. Lawrence Erlbaum Associates, Mahwah, NJ.

Raaijmakers, J. G. W. (2003). Spacing and repetition effects in human memory: Application of the SAM model. *Cognitive Science*, 27:431–452.

Raphael, B. (1976). *The Thinking Computer: Mind Inside Matter*. W. H. Freeman and Company, San Francisco, CA.

Regal, D. M., Rogers, W. H., and Boucek, G. P. (1988). Situational awareness in the commercial flight deck: Definition, measurement, and enhancement. In *Proceedings of the 7th Aerospace Behavioral Technology Conference and Exposition*, pages 65–69.

Reichel, K., Hochgeschwender, N., and Voos, H. (2008). OpCog: An industrial development approach for cognitive agent systems in military UAV applications. In Berger, M., Burg, B., and Nishiyama, S., editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems: Industry and Applications Track*, pages 97–100. INESC-ID, Lisbon, Portugal.

Reyes, J. C., Burgoa, E., Calafate, C. T., Cano, J.-C., and Manzoni, P. (2006). A manet autoconfiguration system based on bluetooth technology. In *Proceedings of the 3rd International Symposium on Wireless Communication Systems*, pages 674–678, Valencia, Spain.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Riley, J. M. and Endsley, M. R. (2005). Situation awareness in HRI with collaborating remotely piloted vehicles. In *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, pages 407–411.

Riskin, J. (2003). The defecating duck, or, the ambiguous origins of artificial life. *Critical Inquiry*, 29(4):599–633.

Ritter, F. E. and Kim, J. W. (2006). Soar: Frequently asked questions list. http://acs.ist.psu.edu/soar-faq/soar-faq.html.

Roediger III, H. L. (2008). Relativity of remembering: Why the laws of memory vanished. *Annual Review of Psychology*, 59:225–254.

Rundus, D. J. (1971). Analysis of rehearsal processes in free recall. *Journal of Experimental Psychology*, 89:63–77.

Ryder, J. M., Weiland, M. Z., Szczepkowski, M. A., and Zachary, W. W. (1998). Cognitive engineering of a new telephone operator workstation using COGNET. *International Journal of Industrial Ergonomics*, 22(6):417–429.

Ryder, J. M. and Zachary, W. (1991). Experimental validation of the attention switching component of the COGNET framework. In *Proceedings of the Human Factors Society 35th Annual Meeting*, pages 72–76.

Salmon, P., Stanton, N., Walker, G., and Green, D. (2006). Situation awareness measurement: A review of applicability for C4i environments. *Applied Ergonomics*, 37:225–238.

Sanford, A. J. and Garrod, S. C. (1981). *Understanding Written Language: Explorations of Comprehension Beyond the Sentence*. John Wiley & Sons, Chichester, New York.

Sarter, N. B. and Woods, D. D. (1991). Situation awareness: A critical but ill-defined phenomenon. *The International Journal of Aviation Psychology*, 1(1):45–57.

Scholtz, J., Young, J., Drury, J. L., and Yanco, H. A. (2004). Evaluation of human-robot interaction awareness in search and rescue. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pages 2327–2332.

Scholtz, J. C., Antonishek, B., and Young, J. D. (2005). Implementation of a situation awareness assessment tool for evaluation of human-robot interaction. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4):450–459.

Schooler, L. J. and Hertwig, R. (2005). How forgetting aids heuristic inference. *Psychological Review*, 112(3):610–628.

Schoppers, M. (1987). Universal plans for reactive robots in unpredictable environments. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1039–1046.

Schweickert, R. and Boruff, B. (1986). Short-term memory capacity: Magic number or magic spell? *Journal of Experimental Psychology, Learning, Memory, and Cognition*, 12:419–425.

Seamster, T. L., Redding, R. E., Cannon, J. R., Ryder, J. M., and Purcell, J. A. (1993). Cognitive task analysis of expertise in air traffic control. *International Journal of Aviation Psychology*, 3(4):257–283.

Sellner, B. P., Hiatt, L. M., Simmons, R., and Singh, S. (2006). Attaining situational awareness for sliding autonomy. In *Proceedings of the 1st Annual Conference on Human-Robot Interaction*, pages 80–87.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, pages 517–524.

Sheridan, T. B. and Verplank, W. (1978). Human and computer control of undersea teleoperators. Technical report, Man-Machine Systems Laboratory, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Shiffrin, R. M. and Cook, J. R. (1978). Short-term forgetting of item and order information. *Journal of Verbal Learning and Verbal Behavior*, 17:189–218.

Sims, C. R. and Gray, W. D. (2004). Episodic versus semantic memory: An exploration of models of memory decay in the serial attention paradigm. In *Proceedings of the 6th International Conference on Cognitive Modeling*, pages 279–284.

Singh, P. (2005). *EM-ONE: An Architecture for Reflective Commonsense Thinking*. PhD thesis, Massachusetts Institute of Technology.

Singh, P. and Minsky, M. (2003). An architecture for combining ways to think. In *Proceedings of the 2003 International Conference on Knowledge Intensive Multi-Agent Systems*, pages 669–674.

Skaggs, E. B. (1933). A discussion on the temporal point of interpolation and degree of retroactive inhibition. *Journal of Comparative Psychology*, 16(3):411–414.

Sloman, A. (2001). Beyond shallow models of emotion. *Cognitive Processing*, 2(1):177–198.

Sloman, A. (2003). Progress report on the cognition and affect project: Architectures, architecture-schemas, and the new science of mind. Technical report, The University of Birmingham. Updated in 2004 and 2008.

Smart, P. (2005). Knowledge-intensive fusion for situational awareness: Requirements for knowledge-filtered awareness. Technical Report DTC/WP270/Requirements, School of Electronics and Computer Science, University of Southampton, Southampton, UK.

Smith, R., Self, M., and Cheeseman, P. (1990). Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, New York, NY.

Sofge, D., Perzanowski, D., Skubic, M., Bugajska, M., Trafton, J. G., Cassimatis, N., Brock, D., Adams, W., and Schultz, A. (2004). Cognitive tools for humanoid robots in space. In *Proceedings of the 16th IFAC Symposium on Automatic Control in Aerospace*.

Sridharan, M., Hawes, N., Wyatt, J., Dearden, R., and Sloman, A. (2007). Planning information processing and sensing actions. Technical Report COSY-TR-0706, School of Computer Science, The University of Birmingham.

Stracuzzi, D. J., Li, N., Cleveland, G., and Langley, P. (2009). Representing and reasoning over time in a unified cognitive architecture. In *Proceedings of the 31th Annual Meeting of the Cognitive Science Society*, pages 2986–2991, Amstrerdam, Netherlands.

Stubbs, K., Hinds, P. J., and Wettgreen, D. (2007). Autonomy and common ground in human robot interaction: A field study. *IEEE Intelligent Systems Magazine*, 22(2):42–50.

Sultanik, E. A., Braude, I., Thai, P., Lass, R. N., Nguyen, D. N., Kopena, J. B., Regli, W. C., Lisse, S. A., Furtwangler, S. N., and Vayda, A. J. (2008). Human-robot collaboration for remote surveillance. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1886–1887.

Sun, R. (2002). *Duality of the Mind: A Bottom Up Approach Toward Cognition*. Lawrence Erlbaum Associates, Mahwah, NJ.

Sun, R. (2003). A tutorial on CLARION 5.0. Technical report, Cognitive Science Department, Rensselaer Polytechnic Institute.

Sun, R. (2006). The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In Sun, R., editor, *Cognition and Multi-Agent Interaction*. Cambridge University Press, Cambridge, UK; New York, NY.

Sun, R., Merrill, E., and Peterson, T. (2001). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science*, 25(2):203–244.

Sun, R. and Peterson, T. (1998). Autonomous learning of sequential tasks: Experiments and analyses. *IEEE Transactions on Neural Networks*, 9(6):1217–1234.

Sun, R., Zhang, X., and Mathews, R. (2006). Modeling meta-cognition in a cognitive architecture. *Cognitive Systems Research*, 7(4):327–338.

Sun, R., Zhang, X., and Mathews, R. (2009). Capturing human data in a letter counting task: Accessibility and action-centeredness in representing cognitive skills. *Neural Networks*, 22(1):15–29.

Taatgen, N. A., Lebiere, C., and Anderson, J. R. (2006). Modeling paradigms in ACT-R. In Sun, R., editor, *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, pages 29–52. Cambridge University Press, Cambridge, UK; New York, NY.

Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., and Schwamb, K. (1995). Intelligent agents for interactive simulation environments. *AI Magazine*, 16:15–39.

The History of Computing Project (2007). Timeline of robotics. www.thocp.net. Downloaded April 6 2009.

Thorndyke, P. W. (1977). Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology*, 9(1):77–110.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press, Cambridge, MA.

Thrun, S., Montemerlo, M., Koller, D., Wegbreit, B., Nieto, J., and Nebot, E. (2004). FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 4(3):380–407.

Trafton, J. G., Bugajska, M. D., Fransen, B. R., and Ratwani, R. M. (2008). Integrating vision and audition within a cognitive architecture to track conversations. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, pages 201–208.

Tulving, E. (1984). Précis of elements of episodic memory. *The Behavioral and Brain Sciences*, 7:223–268.

Tulving, E. (1990). Memory systems. In Eysenck, M. W., editor, *The Blackwell Dictionary of Cognitive Psychology*, pages 222–223. Blackwell Reference, Oxford, UK; Cambridge, MA.

Uhlarik, J. and Comerford, D. A. (2002). A review of situation awareness literature relevant to pilot surveillance functions. Technical Report DOT/FAA/AM-02/3, Office of Aerospace Medicine, Federal Aviation Administration, U.S. Department of Transportation, Washington, DC.

Vaucanson, J. (1742[1738]/1979). Letter to the abbé desfontaines. In *Le Mécanisme du fluteur automate*, page 21. Buren, The Netherlands. translator J. T. Desaguliers.

Venturino, M., Hamilton, W. L., and Dvorchack, S. R. (1990). Performance-based measures of merit for tactical situation awareness. In *Situation Awareness in Aerospace Operations*, pages 4/1–4/5. NATO-Advisory Group for Aerospace Research and Development, Neuilly-Sur-Seine, France.

Weiss, M. D., Peak, J., and Schwengler, T. (2008). A statistical radio range model for a robot manet in a subterranean mine. *IEEE Transactions on Vehicular Technology*, 57(5):2658–2666.

Werger, B. B. (1999). Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams. *Artificial Intelligence*, 110:293–320.

Wickelgren, W. A. (1974). Single-trace fragility theory of memory dynamics. *Memory & Cognition*, 2(4):775–780.

Wickens, C. D. (2002). Situation awareness and workload in aviation. *Current Directions in Psychological Science*, 11(4):128–133.

Wickens, C. D. and Hollands, J. G. (2000). *Engineering Psychology and Human Performance*. Prentice Hall, Upper Saddle River, NJ, 3 edition.

Wickens, C. D., Lee, J. D., Liu, Y. D., and Gordon-Becker, S. E. (2004). *An Introduction to Human Factors Engineering*. Prentice Hall, Upper Saddle River, NJ, 2nd edition.

Wikipedia (2009). Robot. www.wikipedia.com. Downloaded April 6, 2009.

Wilson, N. R., Sun, R., and Mathews, R. C. (2009). A motivationally-based simulation of performance degradation under pressure. *Neural Networks*, 22(5-6):502–508.

Wintermute, S., Xu, J., and Laird, J. E. (2007). SORTS: A human-level approach to real-time strategy ai. In *Proceedings of the 3rd Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 55–60.

Wixted, J. T. (2004). The psychology and neuroscience of forgetting. *Annual Review of Psychology*, 55:235–269.

Wixted, J. T. and Carpenter, S. K. (2007). The Wickelgren power law and the Ebbinghaus savings function. *Psychological Review*, 18(2):133–134.

Yamaoka, F., Kanda, T., Ishiguro, H., and Hagita, N. (2009). Developing a model of robot behavior to identify and appropriately respond to implicit attention-shifting. In *Proceedings of the 4th ACM/IEEE International Conference on Human-Robot Interaction*, pages 133–140.

Yanco, H. A. and Drury, J. (2004). Classifying human-robot interaction: An updated taxonomy. In *Proceedings of the 2004 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2841–2846.

Yoo, J. and Fisher, D. (1991). Concept formation over explanations and problem-solving experience. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 630–636, Sydney, Australia. Morgan Kaufmann.

Zachary, W. and Mentec, J.-C. L. (2000). Incorporating metacognitive capabilities in synthetic cognition. In *Proceedings of the 9th Conference on Computer Generated Forces and Behavior Representation*, pages 513–521.

Zachary, W., Ryder, J., Ross, L., and Weiland, M. (1992). Intelligent human-computer interaction in real-time, multi-tasking process control and monitoring systems. In Helander, M. and Nagamachi, M., editors, *Human Factors in Design for Manufacturability*, pages 377–402. Taylor and Francis, Oxford, UK.

Zachary, W., Ryder, J., Stokes, J., Glenn, F., Mentec, J.-C. L., and Santarelli, T. (2004). Developing concept learning capabilities in the COGNET/iGEN integrative architecture and associated agent-based modeling and behavioral representation (AMBR) air traffic control (ATC) model. Technical Report AFRL-HE-WP-TR-2005-0103, Air Force Research Laboratory, Wright-Patterson AFB, OH.

Zachary, W., Ryder, J., Stokes, J., Glenn, F., Mentec, J.-C. L., and Santarelli, T. (2005). A COGNET/i-GEN cognitive model that mimics human performance and learning in a simulated work environment. In Gluck, K. A. and Pew, R. W., editors, *Modeling Human Behavior with Integrated Cognitive Architectures*, chapter 5, pages 115–176. Lawrence Erlbaum Associates.

Zachary, W. W., Ryder, J. M., and Hicinbothom, J. H. (1998). Cognitive task analysis and modeling of decision making in complex environments. In Cannon-Bowers, J. A. and Salas, E., editors, *Decision making under stress: Implications for training and simulation*, pages 315–344. American Psychological Association, Washington, DC.

Zechmeister, E. B. and Shaughnessy, J. J. (1980). When you know that you know and when you think that you know but you don't. *Bulletin of the Psychonomic Society*, 15(1):41–44.

Zunt, D. (2004). Who did actually invent the word "robot" and what does it mean? http://capek.misto.cz/english/robot.html. Downloaded March 31 2009.