Filtering Data Based on Human-Inspired Forgetting

Sanford T. Freedman, *Member, IEEE*, and Julie A. Adams, *Senior Member, IEEE*

***Abstract*—Robots are frequently presented with vast arrays of diverse data. Unfortunately, perfect memory and recall provides a mixed blessing. While flawless recollection of episodic data allows increased reasoning, photographic memory can hinder a robot's ability to operate in real-time dynamic environments. Human-inspired forgetting methods may enable robotic systems to rid themselves of out-dated, irrelevant, and erroneous data. This paper presents the use of human-inspired forgetting to act as a filter, removing unnecessary, erroneous, and out-of-date information. The novel ActSimple forgetting algorithm has been developed specifically to provide effective forgetting capabilities to robotic systems. This paper presents the ActSimple algorithm and how it was optimized and tested in a WiFi signal strength estimation task. The results generated by real-world testing suggest that human-inspired forgetting is an effective means of improving the ability of mobile robots to move and operate within complex and dynamic environments.**

***Index Terms*—Cognitive science, intelligent robots, mobile robots, robot sensing.**

I. Introduction

ROBOTIC systems are often deployed in complex and dynamic domains where vast amounts of data can flood a robot's sensors. Information overload frequently hinders the ability to detect, recognize, and act on many of the small, but critical, cues present within the environment required for successful task completion. However, humans have an innate ability to navigate through these quantities of data to locate information necessary in achieving high levels of performance and reliability.

While frequently perceived as purely detrimental, forgetting may be a human capability that can aid robots in developing a comprehensive environmental understanding. Forgetting has previously been incorporated into systems that directly select items from memory during action selection [1], but forgetting can also filter information presented to robotic algorithms. This second use of forgetting may purge erroneous and unnecessary data, enabling algorithms to achieve higher levels of performance. This paper presents the use of human-inspired forgetting to act as a filter, refining the data and information presented to existing robotic algorithms. The ActSimple algorithm [2], [3] has been developed as a potential means of realizing this filtering capability.

Manuscript received September 8, 2010; revised March 30, 2011; accepted April 13, 2011. Date of publication June 9, 2011; date of current version November 18, 2011. This work was partially support by the Office of Naval Research MURI award N000140710749. This paper was recommended by Associate Editor E. Santos.

The authors are with the Electrical Engineering and Computer Science Department, Vanderbilt University, Nashville, TN 37235 USA (e-mail: sandytf@alumni.usc.edu; julie.a.adams@vanderbilt.edu).

Digital Object Identifier 10.1109/TSMCB.2011.2157142

This paper provides an overview of the benefits of forgetting and reviews two of its most prominent theories, namely, trace-based decay and similarity-based interference. The novel ActSimple algorithm is presented in Section V. A WiFi signal strength estimation domain used to explore the benefits of forgetting is presented in Section VI. Section VII describes the optimization process used to determine appropriate forgetting parameters. Along with ActSimple, six other forgetting algorithms were optimized for performance comparisons. The results of evaluating ActSimple and the other forgetting methods with real-world data are provided in Section VIII. Finally, Section IX presents a discussion regarding how the results suggest that forgetting and the ActSimple algorithm may benefit robotic systems in improving their cognitive capabilities.

II. Human-Inspired Forgetting

Perfect memory and recall provides a mixed blessing. While flawless recollection of episodic data and procedural rules allows increased reasoning, photographic memory can hinder a robot's ability to operate in real-time highly dynamic environments. Solomon Veniaminovich Shereshevskii, a professional mnemonist, possessed a close to photographic memory, but he found it difficult to learn and understand higher level concepts [4]. Despite the ability to recall lists of great length, understanding nonliteral sentences or recognizing dynamically changing objects, including human faces, proved to be challenging [4]. Shereshevskii's near photographic memory prevented him from forgetting minor details and allowing concepts to transcend into higher levels of thought [5]. A lack of forgetting capabilities can cause similar problems for robots.

The absence of forgetting can fill memory with a tremendous volume of data, increasing 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. Contemporary robots are already overrun with vast volumes of data, and the problem will only worsen. Before robots realize human-level intelligence, a means of classifying the importance of each datum and forgetting unnecessary, erroneous, and expired data will be required. Additionally, removal of undesirable information will aid in generalizing concepts and in achieving higher levels of thought.

Forgetting has previously been recognized as beneficial to machine learning, particularly case-based reasoning [6]. Within some domains and tasks, random forgetting has been found to improve performance [6], and in particular situations, even accurate data can hinder performance [7]. Forgetting may also minimize the occurrence of over-learning. Theories of human

1083-4419/$26.00 © 2011 IEEE

forgetting may provide critical insight into the generation of effective robot forgetting.

Forgetting is a critical cognitive process for humans operating within complex and changing environments. Forgetting can allow humans 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 [8]. Human forgetting can reduce "switch cost," which is the costs incurred when switching between tasks [9]. Through "within-run slowing," which is a performance degradation that occurs during the execution of a task, memories regarding the present task are reduced in encoding strength, allowing easier replacement of task-specific information [9]. Forgetting also aids in classification, which is 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 to incorporate new perceptions, allowing humans to adapt to changing categories [10]. Forgetting also aids heuristic inference. Heuristics play a critical role in the human ability to handle complex decision problems under tight time constraints and demanding mental load [11]. The recognition heuristic states that, when selecting between items, select the item that is recognized [11]. With forgetting, unimportant information is removed, preventing it from being recognized. Irrelevant information removal improves the chances that only important items are recognized, increasing the applicability and performance of the recognition heuristic [5].

III. Theories of Forgetting

Through the years, strong evidence has been found to both support and refute the two leading theories of human forgetting, namely, time-based decay and interference [12]. Some research has posited that human forgetting is a combination of both theories and their subcomponents [12]–[15]. 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 [16]. Others that stand by one mechanism have admitted that multiple mechanisms may contribute to human forgetting, just to a lesser extent [13].

A. Trace-Based Decay

Trace-based decay is the intuitive concept that items within memory deteriorate and eventually disappear due to the effects of time [12]. At perception, items are encoded at a particular activation level, dictating the ease and speed of retrieval. Through recall and reperception, item activation levels increase and slowly decay over time. The memory system can perform a refreshing strategy, where an item is recalled for the sole purpose of increasing that item's activation value [17].

ACT-R is a well-known cognitive architecture employing trace-based decay [1] and is implemented as a production rule system. Declarative knowledge items in ACT-R are called chunks and are selected based on activation levels. Equation (1) calculates chunk activation by incorporating an initial base activation (β) and a logarithmic function operating on item i's J retrieval times $(r_{i,j})$. Decay constant d determines the rate of decay. Source activation (W_k) represents attention directed to portions of the current goal, and strength of association (S_{ki}) represents the frequency that a chunk was required when element k was part of a current goal. ϵ_1 and ϵ_2 represent permanent and transient noise, respectively [1]

$$\alpha_i = \beta + ln\left(\sum^{J} r_{i,j}^{-d}\right) + \sum^{K}(W_k S_{ki}) + \varepsilon_1 + \varepsilon_2 \quad (1)$$

$$P(\text{recall}) = \frac{1}{1 + e^{-\varsigma(\alpha_i - \tau)}}. \quad (2)$$

Chunk selection is nondeterministic. Equation (2) calculates the probability that a chunk's activation exceeds a threshold value (τ) amid noise (ς). If a single chunk activation exceeds τ, then it is selected. When multiple chunks activate simultaneously, the final chunk is selected probabilistically.

B. Similarity-Based Interference

Similarity-based interference, which is the current dominant account of forgetting [12], has a long and storied history [13]. While proponents of interference admit forgetting is correlated with time, they believe that the true causes of failed recall are processes that occur during a time span [17]. Unlike decay theory, interference predicts failed recall results from interitem competition for the memory system's focus of attention [12]. Interference-based forgetting is a complex theory involving many forms of competition, which can be affected by item encoding strength, number of items, item similarity, and phase of learning and recall (encoding, storage, and retrieval) [12]. Two common methods exist for partitioning interference, namely, point of interference (encoding and output interferences) [18] and age of affected items (retroactive and proactive interferences) [12].

The Scale-Independent Memory, Perception, and Learning (SIMPLE) model [19] is a temporal difference model of human memory that places data items in a multidimensional psychological space. A temporal axis records time since item perception. Other axes can exist that store additional properties such as color, weight, or temperature. Item discriminability is determined by the degree in which an item is isolated from its neighbors. When calculating discriminability, interitem distance along the time axis undergoes logarithmic compression [19]. This manipulation increases effective item similarity as items move further from the psychological space's origin. While time plays a critical role in the SIMPLE model, it does

not incorporate trace decay as SIMPLE is purely interference based. In other words, as time passes, items become increasingly similar and difficult to discriminate [19].

Equation (3) computes the interitem distances in the psychological space. Along each axis, the distance $|M_{i,k} - M_{j,k}|$ is scaled by a constant θ_k, representing the level of attention paid to that dimension [18]. The interitem similarity is calculated with (4). λ adjusts how the distance is mapped to similarity, and c is a scaling constant. Item discriminability is generated by (5). 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. When modeling free recall, omissions are possible, and a sigmoid function ((2), with α_i changed to D_i) is used

$$\Delta_{i,j} = \sum^{K} \theta_k |M_{i,k} - M_{j,k}| \qquad \text{where} \sum^{K} \theta_k = 1 \quad (3)$$

$$\eta_{i,j} = e^{-c|\Delta i,j|^{\lambda}} \quad (4)$$

$$D_i = \frac{1}{\sum^{J} \eta_{i,j}}. \quad (5)$$

IV. Robotic Forgetting

Robots are often assigned diverse tasks and are asked to operate within varied domains. No single algorithm or approach will completely solve the entirety of robotic challenges. A large array of algorithms have been developed, which enable robots to move and operate within complex domains while achieving high levels of performance [20], [21]. Unfortunately, these approaches tend to be domain specific. The proposed use of human-inspired forgetting will not replace any existing or future technologies, but it may improve the performance of those systems by allowing more effective searches through the vast quantities of data stored within robotic knowledge bases.

Forgetting and recall mechanisms can select memory items that have a high probability of solving a particular problem. Forgetting can also be applied after a potential memory item has been selected. When memory items cannot be recalled, they are considered unavailable and cannot be used in subsequent calculations. The proposed approach uses forgetting as a filter, reducing the quantity of information available to existing selection mechanisms.

Human-inspired forgetting is a general approach to applying forgetting capabilities to robotic systems, and multiple algorithms can be developed to realize its benefits. ActSimple and the other presented algorithms implement human-inspired forgetting, and all follow the same high-level design paradigm. Upon initial perception, item representations receive properties and statistics facilitating the forgetting process. If memory items are reperceived or recalled, the statistics are updated to reflect current usage patterns. When existing robotic algorithms require items from memory, the forgetting algorithm begins the filtering process. Item retrievability is calculated, and the resultant set of recallable items is presented to the querying system. In general, items are not permanently forgotten as item retrievability is recalculated each time a query is initiated. Depending on the forgetting algorithm used, the existence of memory items, even with a very small chance of recall, can affect item selection, similar to the use of cues to 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. The proposed approach handles both of these cases. Forgetting will not have to be applied to every type of data. In situations where accuracy may be more important than execution speed or when a complete analysis of all memory items is required, filtering can be ignored. The proposed 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.

V. ActSimple

The novel ActSimple forgetting algorithm [2] has been specifically designed to realize human-inspired forgetting in robotic systems. Motivated by numerous cognitive architectures and models of human forgetting, ActSimple incorporates many components, including time-based decay, mental exertion, encoding interference, and output interference. This collection may make ActSimple highly amenable to the varied demands of robots.

ActSimple's **time-based decay component** [(6)] has been heavily influenced by ACT-R. ActSimple does not possess goals, so Act-R's $\sum_k (W_k S_{ki})$ term is not included. Noise is present in ActSimple, but it is located in (12).

Pavlik and Anderson [22] modified (1) to incorporate the item's activation level at each time of retrieval. This modification realizes the spacing effect of short-term memory, and it has been integrated into ActSimple. The d decay factor in (1) was changed from a constant to a value based on activation levels at the time of item recall. $b_{i,j}$ represents the activation level at the jth retrieval of item i, while c_1 and c_2 provide scaling and an offset, respectively. b_i represents item i's current base activation, while $r_{i,j}$ is the span of time since item i's jth recall. A mental exertion scaling factor (φ) was added, and it is described in the following discussion. The initial base level activation β can be set to a fixed value or a dynamically generated value, such as a belief level [21]

$$d_{i,j} = c_1 e^{b_{i,j}} + c_2$$

$$b_i = \varphi \beta_i + ln \left(\sum^{J} r_{i,j}^{-d_{i,j}} \right). \quad (6)$$

ActSimple incorporates **similarity-based encoding interference** to maintain breadth of information. This form of interference is generated through the calculation of discriminability, as in SIMPLE. Unlike the SIMPLE model, however, the acquisition time of memory items is not used to calculate interference as perception and rehearsal-based time effects are already incorporated in the trace-based decay component.

Explicit knowledge items possess properties, either continuous or discrete, that provide interitem 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 to 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 interpret distances along individual axes differently and to dynamically alter their level of importance. ActSimple implements this component with (7)–(9), which are the same as SIMPLE's (3)–(5) except for two changes. A set of δ functions was added to allow task-specific nonlinear transformations along individual psychological space axes, and a set of belief state variables [21] (γ) was also added to represent the robot's belief in an item

$$\Delta_{i,j} = \sum^{K} \theta_k \left|\delta_k(i,j)\right| \qquad \text{where} \quad \sum^{K} \theta_k = 1 \tag{7}$$

$$\eta_{i,j} = \gamma_j e^{-c_3|\Delta_{i,j}|^{\lambda}} \tag{8}$$

$$\wp_i = \frac{\gamma_i}{\sum^{J} \eta_{i,j}}. \tag{9}$$

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 of forgetting an item [13]. As a form of retroactive interference, mental exertion may aid mobile robots in generating focused attention capabilities. When a robot experiences a heavy cognitive load, additional memory items may slow comprehension and may subsequently reduce situational awareness. Mental exertion may result in irrelevant details being encoded at reduced strengths and then quickly removed. Conversely, important items will be recalled rapidly, boosting encoding strength and negating the effects of mental exertion.

Mental exertion has been included with a scaling factor (φ) that modifies initial base activation levels (β). φ is computed by integrating the mental exertion experienced by the robot during a time-window near the item's perception [see (10)]. 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. c_4 acts as a scaling factor

$$\varphi = 1 - c_4 \int_{t_0}^{t_1} MentalExertion\ dt. \tag{10}$$

In complex and dynamic domains, robots experience periods of low computational demand, followed by intervals of high computational load. ActSimple includes **output interference**, which decreases data volume during high load periods. Making decisions increases output interference, systematically removing irrelevant and possibly erroneous information as the rate of decision making increases.

Inspired by Lewandowsky *et al.*'s approach to adding output interference to SIMPLE [18], (11) provides ActSimple with an output interference coefficient ϖ. The ϱ_j term represents the amount of time since decision j was made. c_5 determines the magnitude of interference, c_7 sets the rate of decay, and c_6 provides an offset. Equation (11) incorporates decay. The interference effects of a decision fade as time passes. ϖ is used by (13) when calculating item recallability probabilities

$$\varpi = c_5^{\,c_6 + ln \sum^{J} \varrho_j^{-c_7}}. \tag{11}$$

Equation (12) combines many of ActSimple's components into an **activation level** for each item in memory. The effects of trace decay (b_i) and encoding interference $(\wp_i)$ are summed together with a permanent noise term (ϵ_1) and a transient noise term (ϵ_2). Scaling constant c_8 allows the relative importance of trace decay and encoding interference to be altered

$$\alpha_i = c_8 b_i + (1 - c_8)\wp_i + \epsilon_1 + \epsilon_2. \tag{12}$$

The **recallable probability** equation [(13)] computes that the probability memory items will be made available to an existing robotic algorithm. Output interference factor ϖ reduces item availability when the robot is under periods of decision making. Some forgetting algorithms directly use similar probabilities to select items from memory. ActSimple is different: instead of selecting items, the probability values determine which items are filtered and which are made available to existing robotic algorithms. This change allows existing robotic algorithms to continue performing item selection

$$P(recall) = \left(\frac{\varpi}{\varpi + 1}\right) \frac{1}{1 + e^{-\varsigma(\alpha_i - \tau)}}. \tag{13}$$

Robotic systems may require or benefit from the ability to approximate the ease of memory item retrieval. The ACT-R architecture models memory item **retrieval delays** with (14). A constant latency factor (ψ) scales the result of an exponential function on an item's activation level (α) and a scaling factor (c_9) to generate a retrieval metric. Equation (14) has been incorporated into ActSimple

$$\text{Retrieval Time}_i = \psi e^{c_9 \alpha_i}. \tag{14}$$

In the presented work, a simplified version of ActSimple was employed, which focused on ActSimple's two primary components, namely, trace-based decay and similarity-based interference. Equation (6) was simplified to (15). The λ and γ variables were fixed to 1.0 and were not tested. Mental exertion and output interference were also removed. The activation equation [(12)] was simplified to (16)

$$b_i = \beta + ln\left(\sum^{J} r_{i,j}^{-d}\right) \tag{15}$$

$$\alpha_i = c_8 b_i + (1 - c_8)\wp_i. \tag{16}$$

VI. Evaluation Domain

The effectiveness and practicality of using human-inspired forgetting to filter data were explored by applying ActSimple and a number of other forgetting algorithms to the realistic task of WiFi signal strength estimation. 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, and areas of minimal to no signal strength are frequently encountered. In stable environments, WiFi hardware placement can be optimized to provide acceptable WiFi signal strength while avoiding areas of no communication. Unfortunately, perfect placement of WiFi technology is complex and frequently not possible in dynamic environments. Many robotic deployments involve precision multiagent coordination, which require monitoring and categorization of WiFi capabilities. Some systems even utilize relocatable WiFi basestations to alleviate communication bottlenecks and areas of minimal signal strength. The evaluation domain involved a robot tasked with continuously exploring an environment to monitor and estimate WiFi signal strength. Collected WiFi samples are used to generate signal strength estimations for the entire environment, and interpolation methods were implemented to allow estimation where no WiFi readings were available. This domain is a representative of many activities performed by robots operating where low-level environmental properties can impact the probability of successful mission completion.

A. Algorithms

Seven forgetting algorithms were evaluated to filter unimportant sensor readings: ActSimple, a trace-based decay method (labeled as ACT-R), two similarity-based interference methods based on SIMPLE (referred to as SIMPLE and SIMPLE Update), two queue-based forgetting methods (labeled as 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 in comparing forgetting algorithm performance.

1) ActSimple: The simplified version of ActSimple described in Section V was implemented. The psychological space consisted of three axes: a reading's position along the x-axis, the position along the y-axis, and the WiFi signal strength. Axis selection was subjective, based on the perceived likelihood to benefit performance. The x and y position axes were assumed to hold equal importance.

2) ACT-R: The ACT-R forgetting algorithm is solely a trace-based decay approach, employing the activation mechanism found in both ACT-R and ActSimple. This algorithm was implemented using (15) [changing b_i to α_i to form a simplified version of (1)] and (2).

3) SIMPLE and SIMPLE Update: Both the SIMPLE and SIMPLE Update forgetting algorithms perform similarity-based interference forgetting, inspired by the SIMPLE model of human memory. The same psychological space as ActSimple was used, except that an axis representing time since perception was added. Both algorithms are identical, except when a WiFi reading is reperceived (a WiFi signal strength value is collected multiple times at a particular location). During reperception, the SIMPLE Update algorithm 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 it does not update the perception time. Both forgetting algorithms where implemented using (3)–(5) and (2).

4) Queue Dynamic, Queue Static, and Random Forgetting: Both queue-based forgetting 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, the operation of the two queue algorithms deviated only when a reading was reperceived. If a reading was collected and if it was already present in the queue, then the Queue Dynamic method updated the reading's perception count and moved the reading to the back of the queue. Queue Static also updated the reading's perception count, but it did not relocate the reading.

The Random Forgetting method worked in an identical fashion to Queue Static, except when the queue capacity was exceeded. Unlike Queue Static, the readings were randomly selected for removal.

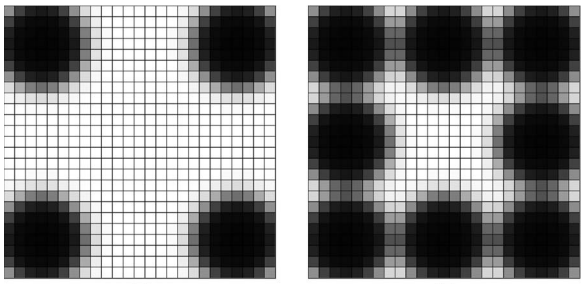
B. Interpolation Method

The forgetting algorithms filtered WiFi signal strength readings that were then processed by an "existing robotic algorithm." During the described experiments, the "existing robotic algorithm" needed to estimate the WiFi signal strength for every location within the test environments. The estimation process placed no restrictions on the paths taken by the robot, resulting in nonuniform sampling and locations in which no samples had been recorded. An interpolation method compatible with nonuniform sample acquisition was used (Nearest Neighbor Interpolation). This fast interpolation method estimates values solely on the closest available sample. In the environment, several readings may be equidistant to the interpolation location. Two separate methods were possible to handle this situation. The first one employs an equal weighting scheme that averages the values for all samples. The second approach creates a weighted average based on the number of times that each reading has been perceived. During the described experiments, the later approach was employed.

The form of weighting used when readings are equidistant can have a large impact on the quality of the 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. 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. If readings with little noise are more common than readings with a larger error, then recall-based weighting provides a greater weighting on the readings with less noise.

VII. Optimization

The forgetting algorithms described in Section VI-A each possess a number of parameters that can greatly influence forgetting performance and estimation accuracy. Before the



(a) (b)

Fig. 1. Tested basestation configurations. (a) Four basestations. (b) Eight basestations.

forgetting algorithms could be applied to real-world data, they required optimization. This section 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 condition was analyzed.

A. Simulated Environment

The simulated environment consisted of a 25 unit by 25 unit grid world, with movements constrained to only horizontal or vertical transitions. At each time step, the simulated robot made a one unit move, unless that position change would cause the robot to leave the bounds of the environment. In this situation, the robot did not move. All sensing and data processing occurred within one time unit. WiFi sensor readings were taken at the end of every time step, irrespective of whether the robot's position actually changed. Signal strength samples ranged from 1 to 100 (weak to strong) and were determined by WiFi basestation proximity [see (17)]. Values computed with (17) were clipped to the range 1–100. s_i represents the distance from the current location to the ith basestation, and ϵ is noise. Two basestation configurations were explored, namely, *four basestations* and *eight basestations* (see Fig. 1). The darker areas in Fig. 1 indicate strong signal strength

$$\text{WiFi Strength} = 100 * \sum_i \left(1 - \frac{1}{1 + e^{-2.0(s_i - 4.0)}}\right) + \epsilon \tag{17}$$

$$p(x) = \begin{cases} 0.62 & \text{if } x = 0 \\ 0.1 & \text{if } |x| = 10 \\ 0.05 & \text{if } |x| = 20 \\ 0.03 & \text{if } |x| = 40 \\ 0.01 & \text{if } |x| = 60 \\ 0 & \text{else.} \end{cases} \tag{18}$$

In addition to basestation proximity, WiFi signal strength readings were also influenced by asymmetric noise. A noise distribution, defined by (18), was applied to the signal readings, and values selected from this distribution were not influenced by noiseless signal strength. This noise distribution provides a high probability of significant outliers, increasing the estimation difficulty. Only nine unique values are provided by the noise distribution, alleviating the need for any confounding binning process to be applied to the readings. The final reading value was clipped to the interval 1–100, converting the symmetric noise distribution into an asymmetric noise source.

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 of effectiveness.

Recorded readings stored the observed WiFi signal strength, along with the position in the environment, time since perception, and perception count. When a signal strength value was observed multiple times for a particular location, separate readings were not stored. Instead, only one reading was generated, and the perception count variable increased. The approach to update the time since perception value was specific to the forgetting method (see Section VI-A). Not all algorithms utilized all data fields within the readings. For instance, some tested forgetting methods did not use the time since perception property while filtering data.

B. Experimental Design

Clustering, distribution, and environment coverage may affect forgetting of low-level data. During optimization, 20 separate paths were used. These paths consisted of three properties: trajectory, basestation configuration, and length. Four hand-generated and four randomly generated trajectories were explored. The randomly generated trajectories were 3000 steps long, but the hand-generated trajectories ranged from 1032 to 3768 steps. Two basestation configurations were present (see Fig. 1), and the basestation configuration changed at the midpoint of six paths. The robot traversed multiple instances of each path. The forgetting methods were applied to each path instance in an offline fashion. This approach allowed all forgetting algorithms to process the same WiFi readings, minimizing the effects of interpath instance variability.

The performance of each forgetting algorithm was calculated at the end of each path. When the robot completed a path instance, it generated an estimated WiFi signal strength value for each location within the environment. Under some situations, the robot was unable to recall any readings for a particular path instance. When this occurred, the robot estimated that each location possessed a signal strength value of 50.5, which is the average of all permissible signal strength values. The estimation error at each location was then averaged to produce a single average WiFi signal strength estimation error value for each instance. The estimation error for each instance was then averaged to form an error metric for each path.

C. Optimization Results

The Queue Static, Queue Dynamic, and Random Forgetting algorithms each possess a single parameter, which is discrete. For these algorithms, an exhaustive search through each parameter space was completed. During optimization, the queue size for each algorithm varied from 1 to 3769. At the largest

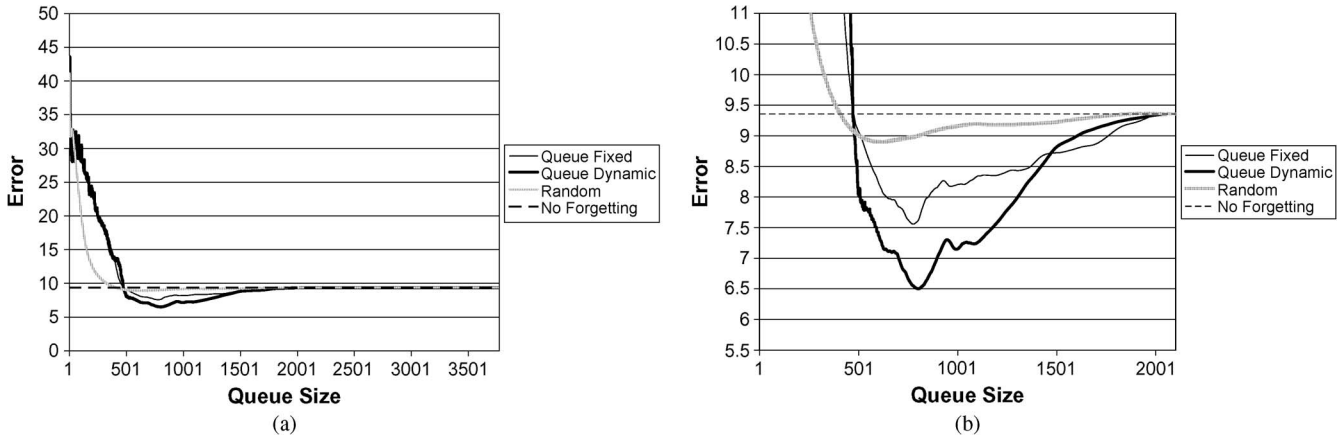


Fig. 2. Queue-based forgetting method optimization results. (a) Full results. (b) Zoomed results.

value, the queue was able to maintain every reading along the path, effectively providing no forgetting. For every queue size, each path was traversed 1000 times. Fig. 2(a) shows the exhaustive search results, while Fig. 2(b) shows 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.

With small queue sizes, the queue-based forgetting methods initially performed poorly (the estimation error is over 25), but the estimation error quickly decreased as the queue size increased (see Fig. 2). Random Forgetting was the first forgetting algorithm that outperforms No Forgetting (a queue size of 409). As the queue size 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 500–1400 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 performances of the three forgetting algorithms become identical to No Forgetting.

Ideally, an exhaustive search would have been conducted through the parameter spaces of each remaining forgetting method. 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 equidistant 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 I lists the number of tested parameterizations for each forgetting algorithm, where the average estimation error was below certain threshold limits. The column labeled as Error Limit provides the threshold values, while the four columns at the right contain the number of parameterizations for each forgetting method that generated estimation errors below the threshold values.

TABLE I
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

Once the parameter sweep was completed, the Coliny Pattern Search algorithm from the Design Analysis Kit for Optimization and Terascale Applications toolkit [23] 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 II 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

TABLE II
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

the performance gains for SIMPLE were negligible (0.001). However, ActSimple received a modest improvement (0.250), and SIMPLE Update's performance unexpectedly improved 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.

The final results from the optimization process, including the results from the queue forgetting algorithms, are presented in Table III. The column labeled as Error presents the estimation error generated by each forgetting condition's optimal parameterization. The columns on the right show the individual parameter values for each optimized parameterization. Parameters not included in a forgetting condition are indicated with a "—" symbol. 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 of 3.113, which is 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.

ACT-R Forgetting'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 path 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 IV.

While the standard deviation between trials for each forgetting method was reduced as the path instance count increased, the values are still relatively small. These results suggest that the forgetting algorithms are robust to differences between path instances.

VIII. REAL-WORLD DATA

Using the parameterizations calculated during the optimization phase, the forgetting methods were applied to real-world WiFi readings. This section describes the testing environment, the method on how the samples were recorded, and the results of processing the data with the forgetting algorithms. This experiment was conducted in an identical manner to the previous simulation, except that a real robot collected the data and new paths and basestation configurations were employed.

A. 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 would be tasked to remove the effects of this interference during testing, uncontrolled interference prevents the generation of truth information that is guaranteed to be consistent and valid throughout the experiment.

Instead, the experiment was performed at the Colson Hollow group campgrounds, which is 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 from the location, but the reliability of the collected truth data was greater than when readings were collected on Vanderbilt's campus.

Similar to the simulation-based experiment, the real-world experiment involved two WiFi basestation configurations and a large number of WiFi sampling locations (see Fig. 3). 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 10 by 10 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 robot was running during data collection, the ground conditions prevented the

TABLE III
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

TABLE IV
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

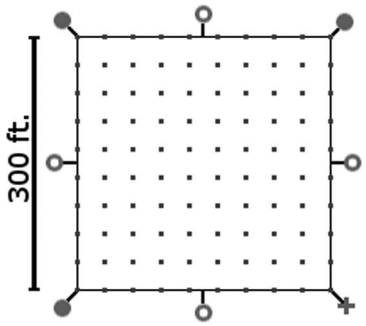


Fig. 3. Real-world experiment map. The points represent the sample locations. The circles represent the WiFi basestations that were always present in both configurations, the rings represent the WiFi basestations present in the seven basestation configuration, and the cross represents a basestation present only in the four basestation configuration.

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 [24] and the NetStumbler 0.4.0 [25] WiFi monitoring program. All samples required for a particular sampling location and basestation configuration were collected at one time for a 6-min interval, with a frequency of up to 2 Hz. The 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 $-32\,618$. These values are erroneous, and they were discarded before processing began. The SNR values were then multiplied by 2.0, incremented by 1.0, and then clipped to the range 1–100. 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, an SNR of over 40 dB is considered to be very good. The clipping process only affects readings with an SNR of 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 prefiltered

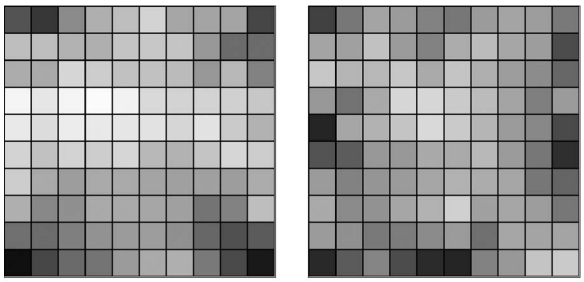


Fig. 4. Maps of real-world truth values. (a) Four basestations. (b) Seven basestations.

data; otherwise, the readings would have been monotonically increasing.

B. Real-World Results

All of the real-world data were 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 VIII-A, the readings for each combination of location and basestation configuration were averaged. The results from this averaging process are shown in Fig. 4(a) and (b). 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 18 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 all forgetting conditions generated effectively identical recall and estimation error values, these paths were removed. The removal of these paths did not affect the error rankings.

Path instances were created by randomly selecting readings from the log files. Readings were allowed to be selected multiple times. During testing, the parameterizations used in the optimization experiment (Section VII) were used.

Table V presents the overall recallable reading counts and estimation error values. 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 conditions 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 conditions averaged at a greater number of recallable readings than ActSimple.

Table VI presents the performance differences between each forgetting method and No Forgetting. 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, 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 a lower estimation error (2.382). Queue Static maintained the third best average estimation error (3.084).

All forgetting algorithms outperformed No Forgetting on 8 of the 18 paths except for ACT-R and SIMPLE Update, which both generated less error on nine paths. The number of paths where the forgetting algorithms underperformed No Forgetting ranged 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 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 conditions. 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 occurred, while SIMPLE Update and ACT-R were the only algorithms that outperform No Forgetting on static paths. However, both forgetting algorithms only outperformed No Forgetting on one path each and by 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. In 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 paths possessing a basestation configuration change versus dynamic paths may decrease.

TABLE V
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

TABLE VI
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

C. Discussion

Forgetting capabilities can potentially improve robotic system performance levels, but they are complex. Just as many factors can influence the performance of existing robotic algorithms, the relative effectiveness of forgetting capabilities may be equally affected. Forgetting involves some risk as even humans occasionally fail to recall important information. However, on average, forgetting benefits humans and may eventually aid robotic systems.

Robotic systems are frequently tasked with operating within complex domains where dynamism can significantly degrade performance. In the real-world experiment, the application of forgetting (excluding SIMPLE) decreased estimation error for every path containing a basestation configuration change. WiFi signal strength is a highly variable environmental property affected by a multitude of sources of interference, including cell phones, computers, humans, and building materials. Even in the controlled environment at Colson Hollow, maintaining low interference levels proved to be challenging. In many realistic environments, including office buildings and even disaster areas, considerable sources of interference will be present, and robotic performance under dynamism may far outweigh static environment performance.

In the real-world experiment, every forgetting algorithm outperformed No Forgetting on some paths but underperformed on others. Except for SIMPLE, each forgetting algorithm's average improvement gain was at least 3.563 times larger than the average performance decrement. These results suggest that, while forgetting capabilities involve some risk, on average, gains will overshadow performance deficits. The forgetting algorithms outperformed No Forgetting when a basestation configuration change occurred (the more difficult situation).

The relative benefits of forgetting depend on the forgetting algorithm, its parameterization, and compatibility with the task and environment. The presented results suggest that benefits from forgetting may not be that sensitive to changing environmental conditions. While optimized for a simulation environment, the forgetting algorithms still successfully reduced the WiFi signal strength estimation error. As many realistic challenges facing robots constantly evolve or are difficult to model, the ability to minimize the need to frequently reoptimize may be paramount in applying forgetting to real-world situations.

Out of all of the forgetting methods tested, the estimation error resulting from ActSimple and SIMPLE Update appears to be the least sensitive to changing conditions. However, in the real-world experiment, ActSimple averaged 293.176 fewer recallable readings. During optimization, ActSimple outperformed all other forgetting algorithms and generated the second best performance during real-world testing. In the simulation, ACT-R generated the third best performance, and Queue Dynamic produced the fourth lowest average estimation error. When operating on real-world data, ACT-R changed to the fifth best forgetting condition, and Queue Dynamic realized the third smallest estimation error.

Additionally, ActSimple averaged the fewest number of recallable readings in the real-world experiment. Robots are real-time systems, and decreasing data processing requirements will aid performance while reducing demands on limited computational and cognitive resources. Forgetting, and ActSimple in particular, may reduce the number of memory items requiring expensive processing by existing robotic algorithms.

IX. CONCLUSION

Forgetting is often considered as a flaw of human memory; however, forgetting provides a critical component in the ability of humans to operate within complex and dynamic environments. Robotic systems may equally benefit from the systematic removal of unnecessary and undesirable information.

This paper has presented a novel approach to incorporating human-inspired forgetting into robotic systems by filtering the data presented to existing robotic algorithms. Inherently non-domain specific, the use of forgetting as a filtering mechanism has the potential to improve the performance and reliability of

many existing and future robotic algorithms. The ActSimple algorithm was developed to realize this forgetting approach, and it consists of a number of components inspired from theories of human forgetting and cognitive architectures.

An optimization process was conducted, which calculated the ideal parameter values for ActSimple and a number of other forgetting algorithms when applied to a WiFi signal strength estimation task. Along with the generation of parameter values, the optimization process provided insight into the relative performance of the forgetting algorithms. Not only did ActSimple generate less error than No Forgetting, but also ActSimple outperformed all tested algorithms.

The forgetting algorithms, combined with the optimized parameter values, were used to process real-world WiFi signal strength data. Despite averaging the least number of recallable WiFi signal strength readings, ActSimple generated the second best overall average error. The results from the real-world experiment suggest that ActSimple is a reliable and stable approach in improving performance through forgetting. The algorithms were optimized in a simulated environment much larger than the real-world environment with different basestation configurations and a simplified noise distribution. ActSimple's stable performance was able to compensate for these differences, providing effective filtering under both conditions. This stability is especially important for robots operating in complex and dynamic domains where environmental conditions are frequently changing and where tasks evolve over time.

Forgetting, and the ActSimple algorithm in particular, has the potential to improve the state of robotics in a diverse range of domains. The forgetting algorithms evaluated in this paper were optimized in the simulation, but many were still able to decrease the estimation error. However, forgetting does entail some risks. In the real-world experiment, the best performing algorithms were able to generate better average values, but they produced inferior results for some paths. It may not be possible to ensure that forgetting algorithms never result in reduced performance. The presented approach in incorporating forgetting into robotic systems allows the filtering effects to be ignored if required by a particular task or domain.

REFERENCES

[1] J. R. Anderson and C. Lebiere, *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates, 1998.

[2] S. T. Freedman, "Human-inspired forgetting for robotic systems," Ph.D. dissertation, Vanderbilt Univ., Nashville, TN, Dec., 2010.

[3] S. T. Freedman and J. A. Adams, "Human-inspired robotic forgetting: Filtering to improve estimation accuracy," in *Proc. 14th IASTED Int. Conf. Robot. Appl.*, Nov. 2–4, 2009, pp. 434–441.

[4] A. R. Luria, *The Mind of a Mnemonist: A Little Book About a Vast Memory*. New York: Basic Books, 1968.

[5] L. J. Schooler and R. Hertwig, "How forgetting aids heuristic inference," *Psychol. Rev.*, vol. 112, no. 3, pp. 610–628, Jul. 2005.

[6] Z. Kira and R. C. Arkin, "Forgetting bad behavior: Memory management for case-based navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 28–Oct. 2 2004, vol. 4, pp. 3145–3152.

[7] S. Markovitch and P. D. Scott, "The role of forgetting in learning," in *Proc. 5th Int. Conf. Mach. Learn.*, 1988, pp. 459–465.

[8] B. A. Kuhl, N. M. Dudukovic, I. Kahn, and A. D. Wagner, "Decreased demands on cognitive control reveal the neural processing benefits of forgetting," *Nat. Neurosci.*, vol. 10, no. 7, pp. 908–914, Jul. 2007.

[9] E. M. Altmann, "Functional decay of memory for tasks," *Psychol. Res.*, vol. 66, no. 4, pp. 287–297, Nov. 2002.

[10] S. W. Elliott and J. R. Anderson, "Effect of memory decay on predictions from changing categories," *J. Exp. Psychol.: Learn., Memory, Cognition*, vol. 21, no. 4, pp. 815–836, Jul. 1995.

[11] ABC Research Group, G. Gigerenzer and P. M. Todd, Simple Heuristics That Make Us Smart, New York: Oxford Univ. Press, 1999.

[12] J. Jonides, R. L. Lewis, D. E. Nee, C. A. Lustig, M. G. Berman, and K. S. Moore, "The mind and brain of short-term memory," *Annu. Rev. Psychol.*, vol. 59, pp. 193–224, Jan. 2008.

[13] J. T. Wixted, "The psychology and neuroscience of forgetting," *Annu. Rev. Psychol.*, vol. 55, pp. 235–269, 2004.

[14] C. R. Sims and W. D. Gray, "Episodic versus semantic memory: An exploration of models of memory decay in the serial attention paradigm," in *Proc. 6th Int. Conf. Cogn. Model.*, 2004, pp. 279–284.

[15] E. M. Altmann and W. D. Gray, "Forgetting to remember: The functional relationship of decay and interference," *Psychol. Sci.*, vol. 13, no. 1, pp. 27–33, Jan. 2002.

[16] S. T. Mueller and A. Krawitz, "Reconsidering the two-second decay hypothesis in verbal working memory," *J. Math. Psychol.*, vol. 53, no. 1, pp. 14–25, 2009.

[17] J. S. Nairne, "Remembering over the short-term: The case against the standard model," *Annu. Rev. Psychol.*, vol. 53, pp. 53–81, Feb. 2002.

[18] S. Lewandowsky, M. Duncan, and G. D. A. Brown, "Time does not cause forgetting in short-term serial recall," *Psychonomic Bull. Rev.*, vol. 11, no. 5, pp. 771–790, Oct. 2004.

[19] G. D. A. Brown, N. Chater, and I. Neath, "A temporal ratio model of memory," *Psychol. Rev.*, vol. 114, no. 3, pp. 539–576, Jul. 2007.

[20] R. Siegwart and I. R. Nourbaksh, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: MIT Press, 2004.

[21] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[22] P. I. Pavlik and J. R. Anderson, "An ACT-R model of the spacing effect," in *Proc. 5th Int. Conf. Cogn. Model.*, F. Detje, D. Dörner, and H. Schaub, Ed.s, Apr. 10–12, 2003, pp. 177–182.

[23] DAKOTA, Design Analysis Kit for Optimization and Terascale Applications, 2010. [Online]. Available: http://www.cs.sandia.gov/dakota

[24] Dell TrueMobile 1150 Mini PCI Card. [Online]. Available: http://support.dell.com/support/edocs/network/tm1150mp/en/index.htm

[25] NetStumbler 0.4.0. [Online]. Available: http://www.stumbler.net/

Sanford T. Freedman (M'03) received the B.Sc. degrees in electrical engineering and computer science from Worcester Polytechnic Institute, Worcester, MA, in 2003, the M.Sc. degree in computer science from the University of Southern California, Los Angeles, in 2005, specializing in intelligent robotics, and the Ph.D. degree in computer science from Vanderbilt University, Nashville, TN.

He worked at IISI Corporation, North Billerica, MA, from 2003 to 2006 and was a Research Assistant with the Interaction Laboratory, University of Southern California, from 2004 to 2006. His research focuses on forgetting capabilities, situational awareness for robotic systems, human–robot interaction, and embedded systems.

Julie A. Adams (M'91–SM'01) received the B.Sc. degree in computer science and the B.B.A. degree in accounting from Siena College, Loudonville, NY, and the M.S.E. and Ph.D. degrees in computer and information sciences from the University of Pennsylvania, University Park, in 1993 and 1995, respectively.

She is an Associate Professor of computer science and computer engineering with the Electrical Engineering and Computer Science Department, Vanderbilt University, Nashville, TN, where she directs the Human–Machine Teaming Laboratory. She worked in Human Factors for Honeywell, Inc., Morristown, NJ, and the Eastman Kodak Company from 1995 to 2000, and she was an Assistant Professor of computer science with the Rochester Institute of Technology, Rochester, NY, from 2000 to 2003. Her research focuses on distributed artificially intelligent algorithms for autonomous multiple robot coalition formation and the development of complex human–machine systems for large human and robotic teams.

Dr. Adams is an appointed member of the National Research Council's Army Research Laboratory Technical Assessment Review Panel on Soldier Systems. She was a recipient of the NSF CAREER Award.