Discovery of High-Level Behavior From Observation of Human Performance in a Strategic Game

Brian S. Stensrud, *Member, IEEE*, and Avelino J. Gonzalez, *Fellow, IEEE*

***Abstract*—This paper explores the issues faced in creating a system that can learn tactical human behavior merely by observing a human perform the behavior in a simulation. More specifically, this paper describes a technique based on fuzzy ARTMAP (FAM) neural networks to discover the criteria that cause a transition between contexts during a strategic game simulation. The approach depends on existing context *templates* that can identify the high-level action of the human, given a description of the situation along with his action. The learning task then becomes the identification and representation of the context sequence executed by the human. In this paper, we present the FAM/Template-based Interpretation Learning Engine (FAMTILE). This system seeks to achieve this learning task by constructing rules that govern the context transitions made by the human. To evaluate FAMTILE, six test scenarios were developed to achieve three distinct evaluation goals: 1) to assess the learning capabilities of FAM; 2) to evaluate the ability of FAMTILE to correctly predict human and context selections, given an observation; and 3) more fundamentally, to create a model of the human's behavior that can perform the high-level task at a comparable level of proficiency.**

***Index Terms*—Context-Based Reasoning (CxBR), fuzzy ARTMAP (FAM), learning from observation, neural network, poker, template-based interpretation (TBI).**

I. Introduction

LEARNING from observation of human behavior is a skill well mastered by human beings, even as young children. Although not all tasks can be fully learned by merely observing others perform (e.g., riding a bicycle and hitting a baseball), many tasks are, in fact, able to be learned by humans through observation (e.g., driving an automobile). In fact, it can be argued that learning from observation shares some commonalities with experiential learning, in that the observer learns from the experience of others. This provides an interesting opportunity for the training of agents to perform humanlike tasks.

There is and has been significant activity in the area of learning from observation in the last several years. We cover that in Section II. This paper describes an investigation into learning the criteria for *context transitions* by observing a player in a computerized game of strategy. To better understand what we mean by a *context* and a context transition, we first present a brief description of *Context-Based Reasoning* (CxBR), which is an essential component of our approach.

Manuscript received May 15, 2007; revised December 6, 2007. This paper was recommended by Associate Editor P. De Wilde.

B. S. Stensrud is with Soar Technology, Inc., Orlando, FL 32817 USA (e-mail: stensrud@soartech.com).

A. J. Gonzalez is with the Intelligent Systems Laboratory, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816-2450 USA (e-mal: gonzalez@ucf.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2008.922062

A. CxBR and Tactical Missions

Webster's dictionary defines context as "...the whole situation, background or environment relevant to some happening or personality" [1]. CxBR, in turn, defines context as previously mentioned, plus the knowledge and functionality for a context-based agent to be able to appropriately act when in this context. In other words, it contains what the agent needs in order to know what to do when in this context. If an agent can identify the context in which it finds itself, it needs only to use the knowledge and functionality defined for that context in order to properly "navigate" it (see [2] for a more detailed discussion on CxBR).

CxBR contexts, in some ways, resemble hierarchical finite-state machines. Indeed, CxBR contexts can be effectively represented by such structures, with contexts roughly equating to states. However, the essential distinction is that contexts in CxBR encompass a grouping of knowledge that is natural (for humans) to a given situation—in effect, anything and everything the agent might need to know while in that context. This knowledge includes functional knowledge (e.g., how to do something) as well as transitional knowledge, which allows the agent to select the most applicable context in a constantly changing environment.

CxBR is an organizational concept and not a language. Contextual knowledge can take the form of one or more functions, rules, neural networks, and simulations, or some form of declarative knowledge. This can drastically prune the search space when looking for relevant operators to address a problem. Grouping knowledge in this fashion can also help in identifying the CxBR context in which the agent finds itself as the transition criteria are defined within each CxBR context (hereinafter called contexts). Traditionally, the contexts have been authored by a knowledge engineer (KE). However, recent research has sought to semiautomatically or automatically build these contexts with the help of knowledge acquisition tools [3] or via machine learning [4], [5]. The work described in this paper is a further effort in the latter approach.

Nevertheless, as the situation experienced by the agent evolves through the natural course of the agent's activity (a game, a mission, a task, etc.), a new set of knowledge may need to be brought to bear ("*activated*") to successfully define and control the behavior of the agent in this new situation. Therefore, recognizing what causes a situation in the environment to change and react to that change by activating the newly appropriate context is not only important but also essential if a system is to correctly perform a behavior. We refer to the criteria that trigger context transitions as the *context transition criteria*. Learning these transition criteria through observation of human performance is the specific objective of the work described in this paper.

We limit our work to problems that involve tactical behaviors. This includes military missions but could also involve team or individual games and other nonconflictive situations where tactical behavior is employed (e.g., driving a car to the airport). The term *tactical behavior*, which is often reserved for behaviors involving military operations, is defined here to denote behaviors with four characteristics.

1) Having a well-defined goal or *mission*.
2) Being characterized by planning and/or maneuvering.
3) Not being well defined as to their execution sequence. Thus, their characteristics may vary greatly across individuals.
4) Needing to intelligently react to unforeseen events or to the actions of others.

B. High-Level Behaviors

The overall behaviors learned by our system are considered to be *high-level* behaviors. The precise definition of a high-level behavior is usually omitted in the relevant literature in spite of the fact that their implementation is a primary focus of the work described therein. Jones *et al.* [6] and Jones and Laird [7] refer to high-level behavior when describing the TacAir-Soar system but never explicitly define the term. Likewise, the work reported by Patterson *et al.* [8] describes a method for learning high-level behavior by examining low-level sensors but also stops short of providing a definition of high-level behavior. A common thread found in all of the literature, however, is that the presence of subbehaviors composes the high-level behavior described. In the paper by Jones *et al.* [6], the behavior of piloting a fixed-wing aircraft is described in terms of the composition of its lower level functionality, such as communication and maneuvering the plane.

In the context of this research, we define high-level behaviors as behaviors that can be represented by a sequence of simpler identifiable subbehaviors known as *low-level* behaviors. A low-level behavior is considered to be *atomic* if it cannot be decomposed any further. Otherwise, between high-level behaviors and atomic behaviors at each extreme, there can be several layers of varying levels of behaviors. For example, in the domain of automobile driving, a high-level behavior could be "driving an automobile." Conversely, "pressing down on the accelerator" is considered an atomic behavior. In between, there are such behaviors as "managing traffic lights," "driving in urban areas" (which could indeed include managing traffic lights), "passing," and "turning left."

If it is assumed that each low-level behavior (atomic or not) can be modeled and identified *a priori*, learning is then the process of identifying and remembering the cues (environmental or otherwise) that trigger the transitions between low-level behaviors. The sequence of these low-level behaviors then composes the high-level behaviors executed by the observed human.

We are, furthermore, interested in a class of low-level behaviors that 1) can be identified during observation; 2) exist *a priori* and need not be learned (only recognized); 3) no two such behaviors can be executed at the same time; and 4) are known to be characteristic of the higher level behavior that we do wish to learn to compose.

Behavior B_i, therefore, is learned by determining how our observed human decides to make use of subbehaviors $b_0, b_1, \ldots, b_k$ that compose B_i. Thus, behavior B_i is considered the high-level behavior. The predefined contexts that compose that behavior therefore reflect the low-level behaviors $b_0, b_1, \ldots, b_k$ that together compose B_i.

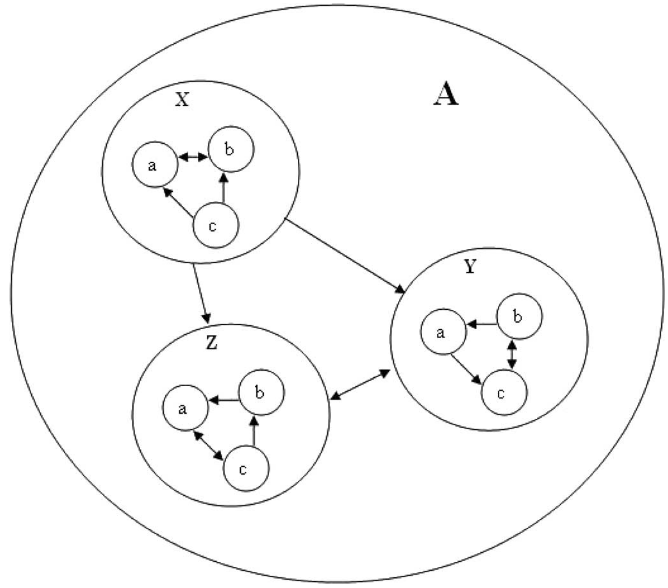

Fig. 1. Learning behaviors by mapping relationships between known subbehaviors.

C. Example of High-Level Behaviors

For clarification on our definition of high-level and low-level behaviors, consider the example where behaviors X, Y, and Z are each composed of a set of known lower level behaviors a, b, and c. The different sequences in which a, b, and c are executed in each high-level behavior serves to distinguish them from each other. Our system learns how a human executes behaviors X, Y, and Z (individually) by creating a mapping between the observations of the human's actions and the sequence of the subbehaviors (a, b, and c) that comprise each behavior X, Y, and Z. Assuming that this task is successfully done, an even higher level behavior A can thereafter be learned in the same manner, provided that its execution is composed of a sequence of behaviors X, Y, and Z. A diagram illustrating this point is provided in Fig. 1.

Behaviors a, b, and c are considered to be low-level (in this case atomic) behaviors with respect to behaviors X, Y, and Z. In turn, X, Y, and Z are considered as (nonatomic) low-level behaviors with respect to A.

These types of situations are easily found when we consider tactical human behavior. The task of flying an airplane, as another example, can be broken down into, in the most extreme case, trivial atomic actions—pushing buttons, guiding a control stick in a certain direction, pushing or pulling on the throttle knob, etc. However, flying an airplane is certainly NOT a trivial task. The real knowledge is contained in the processes involved in deciding when to push a particular button, when to pull back on the stick, etc., and in what sequence, depending on the situation at hand. The knowledge is so complex, in fact, that there are hierarchies of subbehaviors that play a role in representing the behavior of flying a plane. Learning to fly is not achieved by learning "buttonology" or stick-maneuvering techniques per se.

It is achieved by learning to execute procedures (e.g., landing, taking off, and maintaining a heading) that involve knowing when to push what button and when and how to maneuver the control stick and/or the throttle.

The argument posed by this example is that, if given the low-level (atomic or not) functionality used by the human, learning his behavior becomes an exercise in identifying a mapping between environmental and situational cues, which we will call *expert stimuli*, and the low-level function or behavior that the human chooses in response to that cue.

D. *Observations of Human Performance*

In this paper, we describe a learning system that gathers a sequence of observations made of a human performing a high-level behavior. By examining the observations, our system aims to correctly identify the low-level behaviors being executed without feedback from the human and map them to the stimuli within the observations that prompted their selection. With the help of the CxBR modeling paradigm, this system can then be used to develop intelligent models of the learned high-level behavior.

Using CxBR, low-level behaviors are represented as individual contexts, whereas the highest level behavior to be learned is considered to be a CxBR mission. Contexts may contain one and only one behavior (atomic or otherwise) or be composed of several behaviors (atomic, nonatomic, or a combination thereof); which of these is true depends on the context. Some contexts permit only one action to be performed by one atomic behavior. Other situations, however, call for a context that includes more than one behavior although not concurrently.

We define a single *observation* to be a point acquisition of time-dependent inputs used to infer assertions about an agent's environment. We can use time to differentiate and make relationships between two otherwise independent observations. In the following equation, we define an observation $O(t)$ that occurred at time t:

$$O(t) = \langle i_1, i_{2,}, i_3, \ldots, i_n \rangle.$$

Vector $O(t)$ contains fields that represent each input that was introduced to the observer at time t. An observation sequence, therefore, can be considered to be the set of all observations occurring within an arbitrary period of time. The assumption made here is that observations within a time interval occur in discrete points in time rather than continuously. Thus

$$O\{t_0 - t_n\} = \{O\{t_0\}, O\{t_1\}, \ldots, O\{t_n\}\}.$$

As it pertains to our investigation, a single observation includes information about the current environment as well as the current actions of the human. This is critical, because we are attempting to draw a cause–effect relationship between occurrences in the environment and the actions of the observed human. For this research, the learning system develops tactical knowledge from an observation sequence by creating a mapping between an observation pattern and the observed human response. However, it is necessary to process these observations and, from them, learn the knowledge that produces these relationships between the environment and the reaction(s) of the observed human. If we consider these observations as a set of training examples, learning then can be used to generate a knowledge base about actions within the given scenario. Khardon [9] infers a similar definition in his discussion on supervised learning. In our case, however, the learning is to be unsupervised at the input. The observed human does not at all interact with the agent, and learning is done by merely inferring how the human has reacted to his observations. Nevertheless, we define learning from observation as follows:

> *The use of data acquired, through observation, to assert knowledge from which a human's behavior can be intimated.*

We can use our earlier definition of observation to formalize this definition. To do this, we consider the learning process for human E as some function λ of a given observation sequence O_E, i.e.,

$$\lambda\{O_E\} = A_E | A_E = \{A_1, A_2, \ldots, A_w\}.$$

In the preceding equation, the learning algorithm designated by λ operates on an arbitrary observation sequence O_E and outputs a set of assertions A_E that, in some fashion, describe the behavior that has been observed. As the abstraction of "learning" does not imply a restriction in the format of what is learned, these assertions are likewise free to take on various forms: equalities, thresholds, rules, etc.

The potential utility of such a system is twofold. On one hand, the time required to develop acceptable representations of tactical behavior for such agents could be significantly reduced. Instead of producing a complete high-level behavior model by hand, this system could automatically generate what is arguably the most difficult portion of the knowledge: the context transitions.

The second benefit includes the correctness of the knowledge learned. Eliminating a middle person in the development process would conceivably eliminate a source of errors. Furthermore, humans who perform their task with a high degree of proficiency often cannot articulate their knowledge to a third party [10]. A model constructed using a human's introspective explanation can therefore suffer from incompleteness (or even incorrectness) based on this shortcoming. In allowing a system to automatically learn this behavior by observing a human in action, the intermediate step of asking the human to articulate his knowledge is eliminated.

There are, however, some potential caveats in our approach. One is that all contexts and corresponding templates used must be authored *a priori*. This is one significant disadvantage faced by a future developer of an application using this approach. While this is part of the larger problem of knowledge acquisition and machine learning, it nevertheless is quite pertinent to our approach. This paper can indeed serve to reduce the human effort by automatically learning the context transitions. However, significant manual labor is still necessary to prepare the table, so to speak, in order to learn these (e.g., prepare the simulation, run the human subjects, and collect all the observed data). Furthermore, behaviors not predefined as templates cannot be recognized and therefore cannot be learned. These issues are further discussed in succeeding sections.

Before describing our work in greater detail, let us first review the state of the art to see how our work relates to that of others in the field. Given that our application is to poker, we review some of the classic literature on board games and computers.

II. Related Work

Much research can be found in the literature describing learning from observation. While some works address learning high-level behaviors, most focus on learning low-level behaviors. This section describes prior research related to our work.

Board games and computers have a long history together, dating back from the works of Shannon [11], Turing [12], and Newell *et al.* [13]. Charness [14]–[16] studied bridge and chess to identify expertise and their relation to cognitive science. He and his colleagues more recently have used this platform to examine the effects of aging [17]. Certainly, a landmark in computer intelligence was achieved when Deep Blue beat chess Grand Master Garry Kasparov in a chess match in 1997 [18]. This was preceded by important chess playing computers such as HITECH, MEPHISTO [19], and Deep Thought [20], which, prior to Deep Blue, were generally considered to be the best of the chess programs.

Two early researchers of GO playing programs were Zobrist [21] and Ryder [22]. While their work met with partial success, the results of their work could not play as well as a human novice. Additional early work on GO was reported by Kierulf and Nievergelt [23], Kierulf [24], and Wilcox [25].

More to the point, machine learning and board games also have a greatly intertwined history, dating back from Samuel's seminal paper on learning to play checkers [26] and Waterman's subsequent paper on learning heuristics in draw poker [27]. These two seminal works pioneered the machine learning field. Michalski *et al.* appear to be the first to mention observational learning in [28]. Here, they associate learning from observation with unsupervised learning.

In the neural network community, "learning through observation" means that the training data are observations. Fernlund *et al.* [5] define learning from observation as "the adoption of behavior ... through the use of data collected by means of observation." A more descriptive definition describes learning from observation as "inferring concepts by observation" [29]. Here, observation is defined as the act of collecting "characteristics of the relevant environment" [29]. What an observer infers from these observations, however, is a far more complex matter, and so there must be a clear distinction between what is observed and what is inferred about a given environment. One cannot assume that what is reported by a human as "observed" constitutes knowledge that has not already been asserted based on his *a priori* knowledge about his task or scenario. The goal for our learning agent is to develop inferences about "what it sees" based on how a human *reacts* to his observations—not how the human *reports* them. Therefore, observation must be considered as it pertains to the agent—We want to record what the agent sees through the human's eyes. The observations must not, however, include expressions of what the human may annotate or report about his environment.

Sammut *et al.* [30] and Camacho [31] developed systems to observe a pilot's behavior on a flight simulator and implemented the knowledge learned from observation in decision trees. A set of rules was developed as part of the learning process. As part of his work, Sammut coined the phrase "behavioral cloning" to reflect this approach. Sammut's work involves learning rules to perform motor skills involved in flying an airplane. The resulting system learned to fly an airplane as if it were on autopilot in a very strictly defined flight plan. It did not leave room for generalization. Isaac and Sammut's subsequent work [32] extended the previous work to incorporate significant generalization, albeit in a still rather confined domain (maneuvering an aircraft through turbulence).

Sinai and Gonzalez [4] introduced a framework for learning implicit human knowledge through observation of automobile driving behavior within a simulation. Their work is quite relevant to this research because of their attention to partitioning the knowledge by situation (although not called contexts therein). Our work presents almost the opposite approach, in that we assume that the low-level behaviors such as those learned by Sidani and Gonzalez' system (denoted as primitive' in their paper) have already been defined *a priori*. This leaves the actual *situation identification* knowledge to be learned through our neural network approach.

Henninger [33] describes a neural-network-based system that learns how to accurately predict the movement of vehicles in a distributed simulation (ModSAF). Her model builds a predictive model for tank actions by observing a nonhuman but independent algorithm manipulate the tank agent in ModSAF. Gerber [34] employs a *template-based interpretation* (TBI) engine that predicts tank-position information by first selecting its inferred behavioral context. TBI is a method of inferring tactical intent and is likewise essential to our work. It is described in Section III-A. While confined to tank-driving behaviors, Gerber's work is highly relevant to our research. He decomposes the behavior into a set of contexts, which are represented using TBI templates, and using a learning algorithm, he attempts to optimize the identifying weights associated with the templates. The data used in learning is collected from observation of a human-controlled tank. By contrast, the work described in this paper assumes an accurate definition of a set of context templates and attempts to learn the cues that result in a specific context selection.

Johnson *et al.* [35] describe a fuzzy ARTMAP (FAM)-based system that allows computer-generated forces to gradually learn behavior online during a real-time simulation. FAM is reported to have several benefits, including relatively few parameters and the ability to extract and easily explain the results of the learning [36]. FAMs are also essential to our approach.

van Lent and Laird [37] outline the development of KnoMic, a system that extracts knowledge from an expert through observation and then generalizes this knowledge in the form of rules that can be used by an agent to perform a similar task to that of the expert. Whereas Henninger's and Sammut's earlier work focused on learning atomic behaviors from observation, KnoMic is assigned to learn how to execute specific and detailed tasks, such as flying an airplane to a certain destination and in a certain fashion. The authors refer to these types of tasks as performance tasks. As follow-up research to van Lent's KnoMic system, Konik and Laird's work [38] involves the learning of goal hierarchies using inductive logic programming. In the observation mode of this algorithm, the human is again asked to execute a task while annotating goals that he/she has completed during the task. The learning algorithm is then responsible for learning the selection and termination conditions of each goal (when the behavior to execute each goal should be turned on/off). Their use of the human actor beyond demonstrating his skills on a simulator makes their work fundamentally different from ours.

Fernlund *et al.* [5] succeeded in building a system that learned both the low- and high-level behaviors involved in driving a car by observing a human drive a car simulator through a virtual city. Their work used genetic programming to learn individual contexts. Their system generalized quite well and required no intervention by the human actor in the process, beyond performing the behaviors.

Schaal [39] makes a slight distinction between "learning from observation" and "imitation learning." In most cases, learning systems for robots in manufacturing applications try to imitate the exact movement of the human, rather than learn a general behavior. This is typically because, in such applications, the objective of the robot is to imitate the human as closely as possible in a controlled environment.

Walczak and Fishwick [40] describe a study to characterize human expertise by observing the move patterns of chess players. Based on the chunking theory of learning [41], they examine the records of games played by prominent chess masters and a developing player, and compare the chunks learned by these individuals. Their primary objective is not to learn to play the game but to quantify and describe expertise in chess.

Other related work reported in the literature includes that of Pomerlau *et al.* [42], Bentivegna and Atkeson [43], Moukas and Hayes [44], Yang and Asada [45], Floreano and Mondada [46], Pentland and Liu [47], Fogel *et al.* [48], Morrison [49], Crowe [50], Friedrich *et al.* [51], Kaiser and Dillman [52], Rajput *et al.* [53], Hieb *et al.* [54], Gingrich *et al.* [55], Hovland *et al.* [56], Kosuge *et al.* [57], Lee and Chen [58], [59], Khardon [9], Modjtahedzadeh and Hess [60], Fix and Armstrong [61], and Nechyba and Xu [62], [63]. Space limitations prohibit further discussion of these contributions.

Our work differs from the aforementioned works in two ways.

1) We specifically learn the context transitions that are used to link together low-level behaviors into one high-level behavior.
2) We do not interrupt or otherwise consult with the human actor, before, during, or after the learning session. This has the advantage of being able to conceivably learn the behaviors of human actors who do not wish to cooperate with the process (e.g., an opposing team and military enemies). We discuss this in more detail in Section VI.

The works closest to ours is that of Konik and Laird [38] and van Lent and Laird [37] in that they both learn high-level behaviors. However, consultation with the human actor appears to be essential in their approach. Our work represents a different approach to the work of Fernlund *et al.* [5]. Whereas they learn the low-level contexts as well as the transition rules, our work concentrates on learning the transition rules using a vastly different approach.

III. Our Approach to Learning From Observation

Here, we describe an algorithm that identifies low-level (possibly atomic) behaviors when executed by the human and creates a mapping between them and the observations that precede them. The name of this algorithm is *FAM/Template-based Interpretation Learning Engine* (FAMTILE). However, brief descriptions of TBI and FAM neural networks are provided for the interested reader. Readers familiar with these techniques can skip to Section III-C.

A. Template-Based Interpretation

TBI was conceived by Drewes [64] and later enhanced by Gerber [34]. TBI infers tactical intent from observed atomic actions and allows for an inference to be made about the low-level sequence of actions executed by the human and observed by our system. In TBI, contexts are represented by *context templates* or *templates*, which list the expectations of what a human would have to do (in terms of atomic actions) when in the process of carrying out the intended actions. By progressively checking off as "done" the actions that are actually observed, a clearer picture of the intentions of the observed actor comes into focus. Within each template is a set of attributes that indicate actions and conditions; each attribute within a template is considered to be relevant to the context represented by that template. TBI operates by associating a specific observation or observation sequence to the attributes of each template to determine which (if any) of the attributes are satisfied. TBI continuously computes a cumulative score for each template over time. This score is proportional to the number of attributes of a template that are satisfied (Drewes called it "checked off" in his dissertation [64]) and their respective weight. As time passes and more observations are logged and compared to the template's attributes, the cumulative scores of those templates that, in fact, reflect what is happening will tend to rise, whereas those that are irrelevant will either remain low or possibly decrease. At a certain point in time, the template earning the highest score is flagged by the TBI engine as having sufficient confidence that that context is indeed what the observed performer is doing. This process resembles the game of Bingo in many ways. A card is analogous to a template, and a number call to an observation. When a threshold is reached in a specific card (a horizontal, vertical, or diagonal line is checked), success can be declared by yelling "Bingo."

As an example, consider the tactical behavior of driving a car. As a high-level behavior, driving includes several lower level behaviors executed in support of the high-level task: stopping at a red light, passing slower traffic, avoiding and being aware of pedestrians, etc. Oftentimes, there are attributes and cues from the driver and/or from the surrounding environment that can indicate to an observer which atomic behavior is being executed by the driver. For instance, a passenger does not need to ask the driver to indicate when he's attempting to pass a slower car, he can simply look out the window—the driver has changed lanes and increased his speed, the passed car is driving too slow, etc.

In TBI, we consider these cues to be the attributes of a context and group them together within a context template. These attributes are then assigned a weight indicating their importance in identifying the context. Because the behavior expected within each context is known *a priori*, creating templates with useful attributes is a reasonable task for a KE.

B. FAM Neural Networks

FAM is a neural-network clustering technique developed at Boston University in the early 1990s. The network was introduced by Carpenter *et al.* [36] and is described in detail by

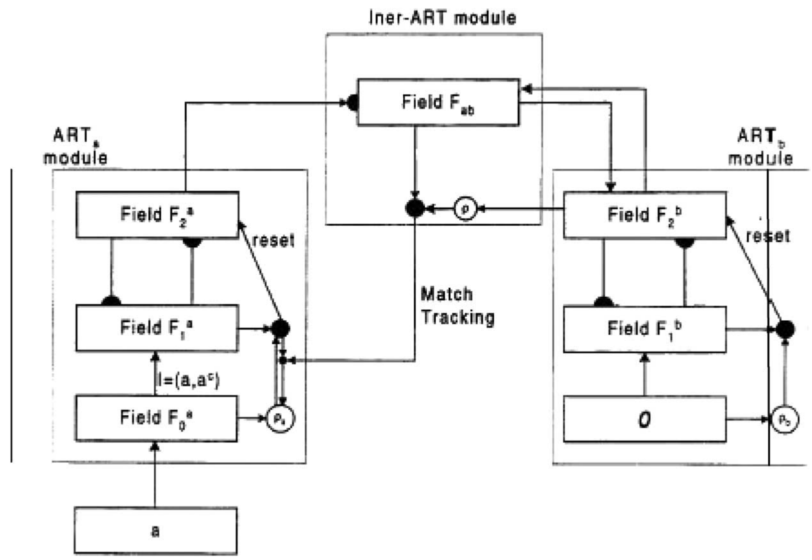

Fig. 2. Block diagram of a FAM architecture [65].

Georgiopoulos and Christodoulou [65]. The goal behind this technique is to produce a neural network that is proficient at dealing with "misbehaved" batches of test patterns, i.e., patterns where a minority of the testing patterns share little in common with the majority used to train the neural network but are equally (if not more so) relevant.

A block diagram of the FAM architecture is provided in Fig. 2. The ART$_a$ and ART$_b$ modules within FAM are responsible for generating pattern templates that correspond to a certain pattern form, essentially dividing the pattern set into clusters. Each template created within the ART$_a$ module represents an input-pattern type that corresponds to a specific output template created by the ART$_b$ module. The Inner-ART module is then responsible for creating a many-to-one mapping between the templates within ART$_a$ and those within ART$_b$.

For example, consider a situation where a neural network is trained to recognize alphabetical letters when seen and, in response, produces a specific sequence of numbers based on the letter input. When training a FAM module, the ART$_a$ module is responsible for learning to recognize each input letter, whereas the ART$_b$ module is responsible for learning to recognize each output sequence. The Inner-ART module creates the mapping between specific letters and their corresponding output sequence.

C. *Our Approach*

The FAMTILE algorithm is composed of two major parts: Part 1 involves inferring the context being experienced by the human actor being observed. Part 2 relates to mapping the context inferred in part 1 to the environment to determine the potential causes of a context transition. Part 1 employs the aforementioned TBI algorithm, whereas part 2 employs FAM neural networks. These two parts are independently discussed.

After learning the set of conditions that trigger atomic behavior transitions, a CxBR model that reflects the high-level behavior of the human observed during the simulation can then be constructed. This model contains both the low-level contextual knowledge developed *a priori* and the knowledge learned by this system that identifies when each low-level context becomes activated. We begin this section by defining terms and discussing how the observational data are captured.

1) Acquiring the Observational Data: Before the learning process can begin, the human actor to be observed must clearly understand the mission he is to perform. He must also be in an environment (either live or simulated) that he can affect through his actions. Furthermore, the observational system must be situated so it has the most direct access to the stimuli seen by the human actor without impeding him in any way. In this paper, we simplify the problem somewhat by using a simulator to implement the learning algorithm. This facilitates the observation process and allows us to concentrate on the technical feasibility of the algorithm.

While the human actor executes a high-level mission within the simulation, FAMTILE records all relevant and visible stimuli on the human, along with the actions taken by the human at the time those stimuli are presented. A recording is made at each decision point i reached during the execution of the behavior to be learned. In the simulated world, these decision points can be either continuous points or segments of time or planned decision points where time is not relevant, such as in a turn-based game, such as chess or poker. To account for the reactive nature of the human's actions at any decision point i, we refer to the time at which the stimuli are presented as time i^- and the time at which the human switches his active context as time i^+. We assume that the human cannot anticipate the environmental trigger but must perceive it before acting to switch contexts. Anticipation is a complicating feature at this time, and we leave that for future research. However, we see no fundamental impediment to a future implementation of this feature.

At the point when the human completes the scenario, the learning system will have compiled a set of recordings that should encompass all relevant stimuli and the actions taken by the human actor. This set is known as the *observation sequence* for the executed scenario. Individual members of this sequence are distinguished by the simulation-time at which they were recorded and are referred to, naturally enough, as *observations*.

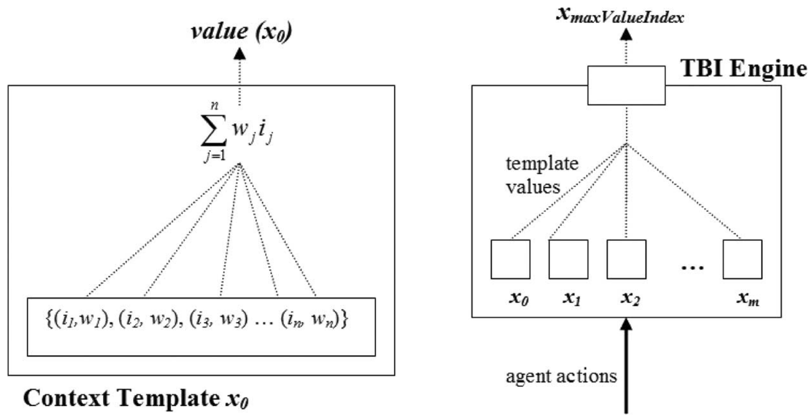

Fig. 3. Generic context template and the TBI engine.

These observations, which are labeled σ_i, denote decision point i, along with the set of visible stimuli Φ that existed at i^- and the set of actions Γ taken by the human at i^+. Thus

$$\sigma_i = \langle \Phi_{i^-}, \Gamma_{i^+} \rangle$$

where $\Phi_{i^-} = \{o_0, o_1, \ldots, o_n\}$ are the traits of observation i, and $\Gamma_i = \{j_o, j_1, \ldots, j_n\}$ represent the actions taken by human in response to observation at i.

We define the complete observation sequence Ω_n to be the set of observations σ_i taken of the human throughout an entire scenario n, i.e.,

$$\Omega = \bigcup_i \sigma_i.$$

After the observations of the human are complete, the entire observation sequence Ω is presented to FAMTILE. At this point, the actions of the human are interpreted by the TBI engine, which will convert Ω_n into a new observation sequence $\Omega_n^{'}$, where the set of actions taken (represented by σ_i in Ω_n) are replaced with the interpreted context. This context, which is inferred by TBI for decision point i, is represented by Ψ_{i^+} in the following equation:

$$\sigma_i^{'} = \langle \Phi_{i^=}, \Psi_{i^-} \Psi_{i^+} \rangle$$
$$\Omega' = \bigcup_i \sigma_i^{'}.$$

In addition, represented within $\sigma_i^{'}$ is the inferred active context of the human prior to decision point i. This context is denoted as Ψ_{i^-} and is identical to the context inferred from the previous decision step Ψ_{i-1^+}. FAMTILE's TBI engine achieves this transformation by making an interpretation of each atomic action. Prior to the observation time, a KE defines each atomic behavior (i.e., the behavior the system will observe) that is necessary for the execution of some high-level behavior (the behavior the system will infer). From these specifications, the KE also creates a set of context templates. Each of the templates' attributes is derived from fields within observation σ_i.

Now we move on to the first part of the FAMTILE process: how to infer the human's context.

2) Part 1—Inferring the Context of the Human Performer: We assume that all low-level behaviors can be identified through observation. Because the low-level behaviors that compose a particular context are known, we need only recognize them through observation and record their presence. Then, we must put them together into a sequence that explains the higher level intentions (i.e., the context) of the observed performer. To accomplish the latter case, we employ the TBI technique discussed in Section III-A.

For convenience, we will consider an arbitrary set of contexts $C = C_1, C_2, \ldots, C_n$ and corresponding set of templates $T = T_1, T_2, \ldots, T_n$. Using this representation, we say that a template T_j includes all attributes and weights common to its corresponding context C_j. In a given scenario, all contexts C_i are represented within TBI by a specific template T_i that defines the attributes of C_i.

Each attribute a_i in template T_j is a representation of a condition that is prevalent in context C_j. Weight w_i represents the importance of a_i in determining context C_j. A low weight value for w_k indicates that attribute a_k is not an essential or even very important characteristic of context C_j. Conversely, a high value for w_m indicates that attribute a_m is highly relevant, perhaps even essential, for context C_j. This representation was used in both the works of Drewes [64] and Gerber [34]. Thus

$$T_j = \{\langle a_0, w_0 \rangle, \langle a_1, w_1 \rangle, \ldots, \langle a_n, w_n \rangle\}.$$

The TBI engine infers a context by first evaluating the *state* of each attribute in its set of predefined templates. After each attribute is assigned a value (typically T or F, depending on whether that action has been observed or not), a weighted sum is computed for each template T_j and used as its *template score*. This template score s_j is computed as follows:

$$s_j = \sum_{i=0}^{n} a_{ij} w_{ij}.$$

The value assigned to each attribute a_i in template T_j depends on the nature of the attribute. Fig. 3 represents a TBI engine that considers a set of m context templates and n attributes per template. On the left side of the figure, we see the composition of a generic context template score. Note that the score is generated using a simple weighted sum of each attribute score (computed using the preceding equations). The right side of the figure illustrates the comparative portion of the engine—each score is reviewed and the maximum score is selected. The context associated with $s_{\max}$ is chosen as the inferred context for that observation. Stensrud [66] provides a more thorough description of how TBI is applied to FAMTILE. The output of this first part, therefore, is an indication of what context the human is experiencing while the system observes his actions.

3) Part 2—Associating Context Change to Environmental Triggers: This section discusses the part of the FAMTILE algorithm that learns the transitions between contexts affected by the human performer. It accomplishes this through neural networks.

The ability of a neural network to handle "misbehaved" training sets is of particular relevance to learning from observation. Consider the knowledge required to drive an automobile, which is an example of a tactical skill. The ability to handle a tire blowout while driving, particularly when at high speeds, is certainly important. However, this skill is rarely required, simply because tires rarely ever blow out. If one were to observe an automobile driver in order to train a neural network how to drive, the training pattern corresponding to a blown-out tire would represent a very small minority of the training set.

In a CxBR model for tactical control of an entity in a simulation, it is possible that important events requiring a specific context transition infrequently occur. Because of this, training patterns representing these types of context transition cues will most likely be underrepresented within a training set. In such situations, traditional neural networks have a difficult time learning these patterns as a result of the strong emphasis on the other patterns. In these cases, the neural network tends to "overlearn" the more frequent patterns and discard the others as noise within the training set. In the case of our work, this noise may represent an interesting and important observation, making the human's response to it very important to record. FAM neural networks are adept at recognizing the infrequent patterns without reversing the knowledge of any well-learned patterns [65].

Through the creation of clusters, FAM also has the ability to handle a large sample of training patterns necessary for a complete observation of a human's behavior. This clustering process has the effect of significantly reducing the complexity of a decision space, based on the size of the clusters created. The advantage here can be visualized by again considering the task of learning driver behavior. Because recording a decision-making cue (e.g., to change lanes, to brake, and to turn) often requires fine granularity across observations, several hundred observations of the driver/expert may be recorded throughout a few-minute driving task. Furthermore, values for the driver's speed, heading, distance to other vehicles, and other potentially significant factors will certainly fluctuate, at least nominally, along a several-second interval where no significant behavioral change is executed. This is not because the driver consciously decides to make these changes (decisions that should be recorded and learned) but simply because of the dynamics of the environment and the driver's inherent inability to hold an identical speed and course. A FAM system allows for nearly identical input patterns such as these (that map to the same output) to be represented by a single cluster. By creating a less complex decision space, we significantly reduce the order of the learning task.

Our specific learning objective here is the transitions between contexts. The new context would contain the appropriate functionality to allow the agent to properly manage it. FAMTILE is built to recognize and capture those triggers and learn them for subsequent use by the agent. We assume that all other functionality—that which permit a context to correctly control an agent when active—is already known *a priori*.

Set Ω' is, at this point, transformed into a form usable by FAM. This operation is done by converting each σ_i' into a single training pattern. For a training pattern to be readable by the FAM neural network, each field must be a *fuzzy value* (some real number between $[-1, 1]$). Within FAMTILE, the input portion of the training pattern is derived from $\Phi_{i=}$ and Ψ_{i-}, whereas the output pattern is derived from Ψ_{i+}.

The subset $\Phi_{i=}$ of observation sequence Ω_n' consists of fields representing the human's complete observation at time i^-. The human's active context at i^- is denoted by Ψ_{i-}. Converting the observation for Ψ_{i-}, the observed active context at i^- involves the same procedure, regardless of the scenario. To convert the identified active context into a field within the input pattern, one field is set aside for every possible context in the scenario. If a context j is identified as the active context, the jth field is assigned a value of 1, and the other "context fields" within the input pattern are assigned a value of 0.

This is done to persuade input patterns with different active contexts to bind to different templates in ART_a. The following equation represents an arbitrary input pattern converted from $\Phi_{i=}$ that can be presented to FAM, which we refer to as $\dot{\Phi}_{i=}$:

$$\dot{\Phi}_{i=} = \overbrace{o_1, o_2, o_3, \ldots, o_{k-1}}^{\text{observation fields}}, \underbrace{c_1, c_2, c_3, \ldots, c_{n-1}}_{\text{active context}(n-1)}.$$

Output pattern Ψ_{i+} is simply a representation of the inferred active context at i^+. Because of this, Ψ_{i+} can be represented as a j-bit binary number to identify one of j distinct contexts as active, just as is done for the inferred context at i^-. Within Ψ_{i+}, all bits are set to 0, except for one. If that one set bit is the ith bit (i.e., oc_i in the expression for $\dot{\Psi}_{i+}$), that means that context i has been identified as the active context for i^+. This representation scheme will make for a trivial clustering task for ART_b, because exactly one output cluster will be generated per context. Representing a context name in this manner allows for the output of ART_b to be both readable and unambiguous for either a KE or a separate module created to read its output. The following equation represents an arbitrary input pattern converted from Ψ_{i+} that can be presented to FAM, which we refer to as $\dot{\Psi}_{i+}$:

$$\dot{\Psi}_{i+} = oc_1, oc_2, oc_3, \ldots, oc_{n-1}$$
$$(\text{a bit string representing the selected active context}).$$

The input and output patterns $\dot{\Phi}_{i=}$ and $\dot{\Psi}_{i+}$ presented to FAM reflect observations recorded at specific times during the scenario, along with the active contexts at those times, as identified by the TBI engine. The input patterns are represented by quantitative values for each stimulus on the human—enemy movements, environmental conditions, current physical conditions, etc. The output patterns represent the action taken by the human in response to the input pattern presented, where each action reflects a transition from the provided context at the input to a new active context which is inferred using TBI. The implication here is that every action (and thus every output pattern) will

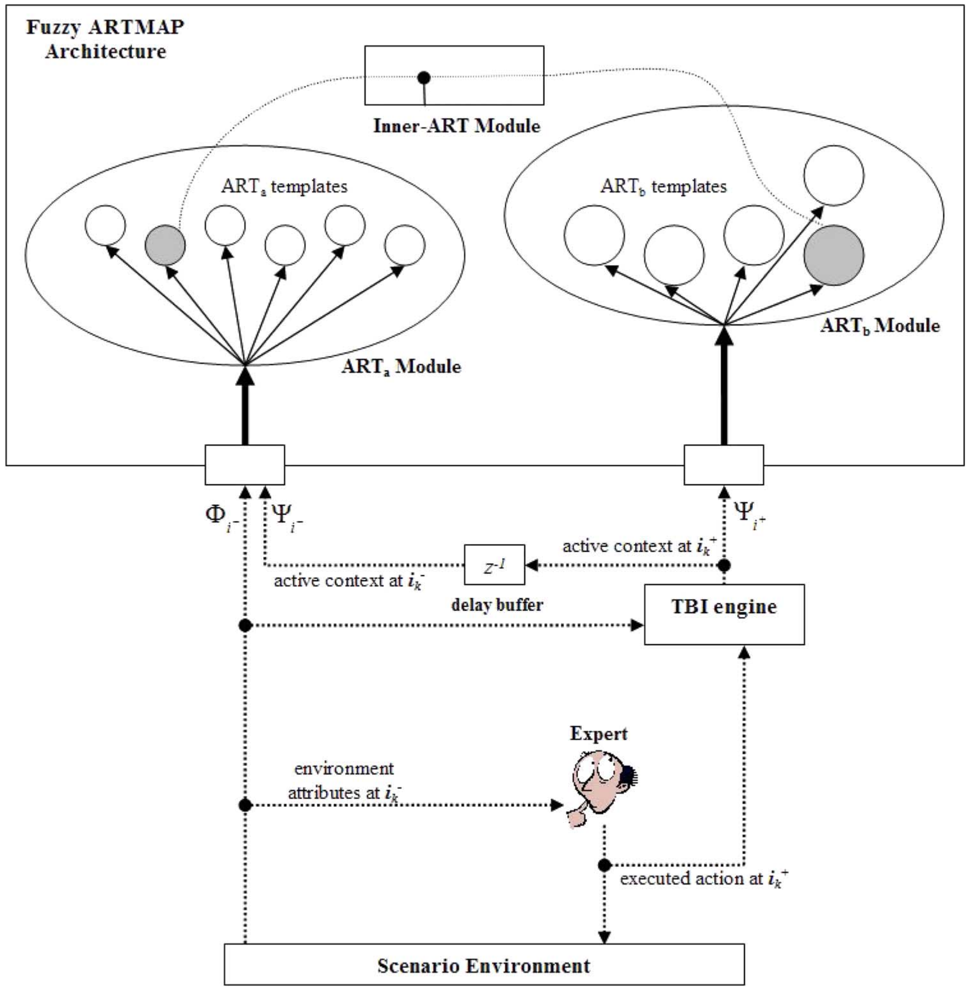

Fig. 4. Learning context transitions in FAMTILE.

represent a transition to a new context, which is of course not always the case. Rather, actions representing no context transition are also represented by patterns that require a transition to the current context—the equivalent of no context change.

A training pattern is generated and presented to FAM for each observation made of the human during the execution of a scenario. Learning occurs through the creation of clusters in the ART$_a$ and ART$_b$ modules and of a many-to-one mapping between those templates. ART$_a$ templates represent clusters of input patterns, similar in their representation, to which the human has responded by making a specific context transition. That transition is stored in a template in the ART$_b$ module, and a mapping between the two templates is created. When the network subsequently encounters an input that matches the input pattern cluster represented by that template in ART$_a$, it will know that the appropriate response is stored in its mapped template in ART$_b$.

Fig. 4 illustrates FAMTILE in learning mode. A recorded observation includes both the stimuli on the human and his resultant decision. A decision is considered to be the action made by the human in response to a set of stimuli presented at i and is expressed as the context that the agent enters (makes active). These stimuli, along with the active context in which the human is operating at i^-, constitute the input pattern that is presented to ART$_a$. The actions that the agent executes in response to these inputs (at i^+) are analyzed by a TBI module, which then outputs the most likely candidate for the context that corresponds to those actions. That context name is then presented to ART$_b$ as the output pattern for i and is also stored for the next decision-point $i+1$, where it will be presented as part of the input pattern as the active context prior to the stimuli presented and actions taken at $i+1$.

The task for FAM, then, is to learn the correct context transition, given the current active context and the input stimuli on the agent. To do this, the network will create templates in ART$_a$ that effectively cluster similar input patterns that induce a specific context transition by the human. The template corresponding to the actual transition made will be stored in ART$_b$, and the Inner-ART module will create a link representing a mapping between the two templates. After the training phase is complete, there will exist a many-to-one mapping between the input-pattern templates in ART$_a$ and the context transition templates in ART$_b$.

D. FAMTILE Operation

A summary of the sequence of events required for the FAMTILE algorithm is presented here.

1) The human actor executes a high-level behavior in some simulation or simulator.

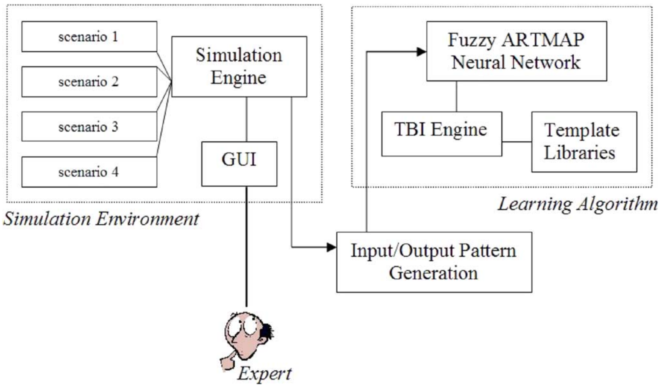

Fig. 5. Block diagram of the testing environment.

2) FAMTILE collects an observation sequence of the human's actions.
3) The TBI engine interprets human actions and infers corresponding contexts.
4) The observation sequence with contexts inserted is converted into a set of input patterns.
5) The sequence of contexts is converted into output patterns.
6) The input/output patterns are paired and presented as training patterns for the neural network.
7) The neural network is trained to recognize observation patterns and map them to specific high-level contexts.

IV. Test Prototype

To evaluate the FAMTILE concept, a prototype system was built. However, in evaluating this prototype, it was first necessary to construct a test bed simulation in which training vignettes could be developed and executed. This simulation was written in Java and was designed to interface the FAMTILE prototype with the testing vignettes and to provide a graphical user interface for the human actor to perform his behaviors. A block diagram of the simulation environment is provided as Fig. 5.

The simulation engine provides both the logic of the vignettes and their graphical user interface, which was developed in Java. This interface was created in an attempt both to attract human test subjects to participate and to provide them with as realistic a vignette as possible.

The simulation engine implements the logic and execution engine for each of the four vignettes. When a human subject selects one of them, the simulation instantiates it and presents the human with his first decision point. Each vignette is such that the human actions are *turn based*, and observations for a certain decision step represent a set of stimuli and resultant action for one turn. In a turn-based simulation, decision steps are triggered on human actions and not on actual clock time. This property ensures for FAMTILE that the human is making decisions in response to a known set of observations and that there is a correct pairing between those observations and that action. Otherwise, the system could not guarantee that the human was making decisions based on the observation recorded for that corresponding time step. The actions that take place within the simulation during training mode are presented here.

- The simulation prompts the human actor to enter his/her name.

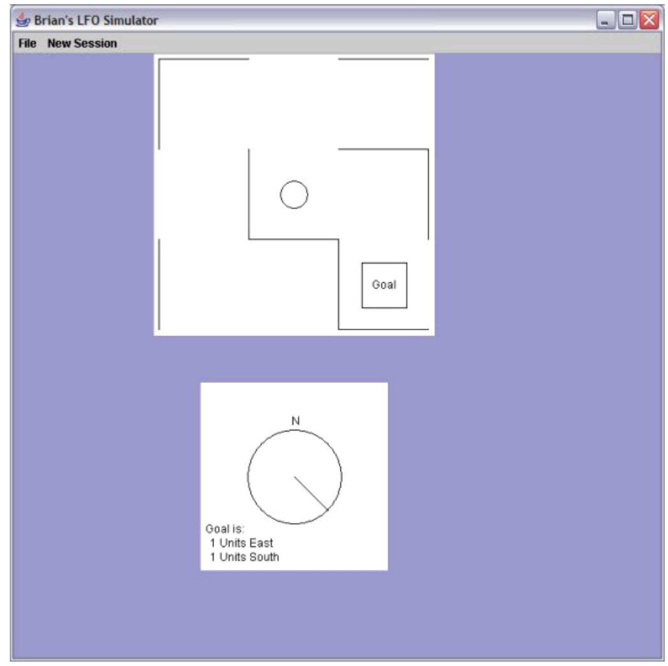

Fig. 6. Vignette A.

- After the name is entered, the human selects a training vignette.
- When a vignette is selected, the simulation engine calls the initial commands that begin that vignette. That vignette then displays the situation for the human and then pauses until the human has made his/her response.
- That response triggers an event in the simulation that brings up the next situation and writes the stimuli/response pair to a text file, which is read by the interface class after the training session.

To make a thorough evaluation of the learning algorithm, four different test vignettes were developed. These are based on two behaviors: 1) moving within a maze environment and 2) playing a game of poker.

A. Maze Navigation: Vignettes A and B

The first two training vignettes involve the navigation of a 2-D maze. For each vignette, the human is asked to navigate from his position within a virtual maze to a specified goal position. At each point during the vignette, the player is provided a compasslike directional icon that indicates the distances—in both the x and y directions—to the goal position. If the goal position is located within the player's field of view, its position is marked on the map.

In Fig. 6, the circular shape occupying the center position in the maze indicates the position of the human's avatar. In vignette A, the player can only see one space in all directions from the avatar's position. From the observations of this figure, the human makes a decision on which direction to move. In this vignette, the avatar and goal positions are reinitialized after each human action.

In vignette B, the human is asked to navigate the avatar toward a goal position and is given a larger frame of view (see Fig. 7). The simulation also records the spaces that have been visited by the avatar along his path to the goal position and marks these spaces with a square shape on the maze view.

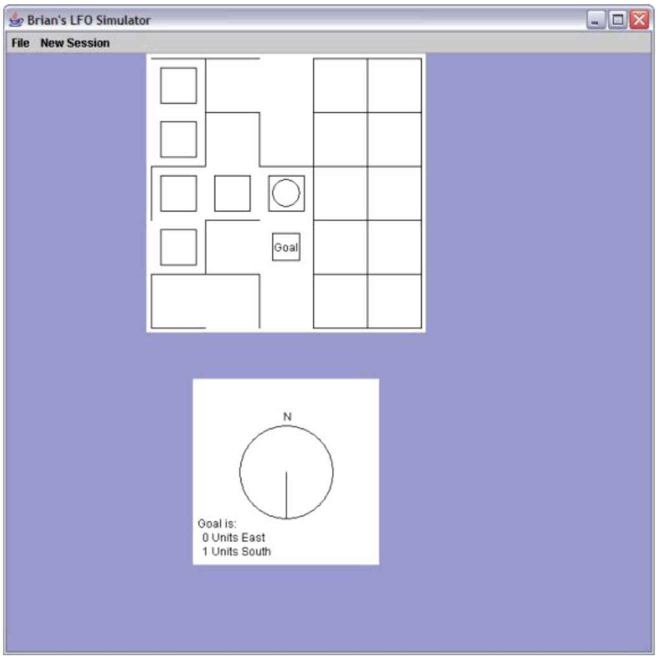

Fig. 7. Vignette B.

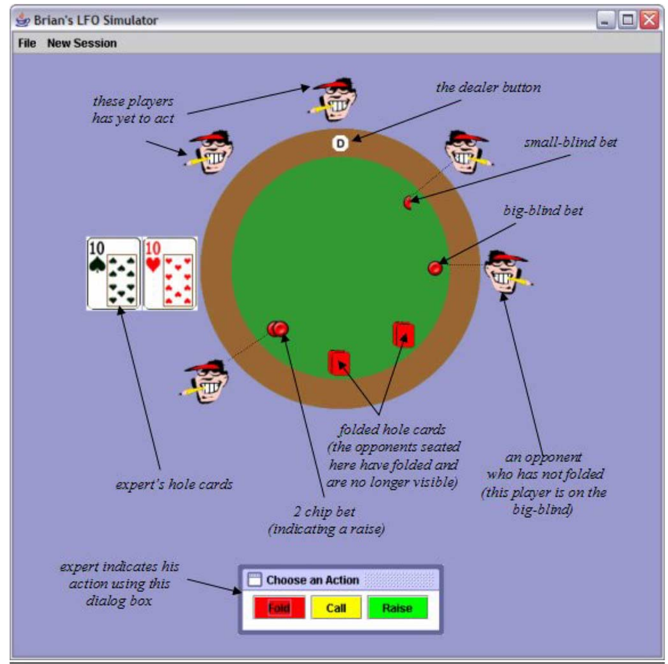

Fig. 8. Vignette C.

For vignettes A and B, no context templates are required, because there are no contexts implied with the human's movement. Vignettes A and B are used to provide control cases to evaluate the ability of the FAMs to learn without the encumbrance of the FAMTILE system. More details on this are provided in Section V.

B. Poker Game: Vignettes C and D

The other two training vignettes involve the game of Texas Hold'em Poker. The succeeding sections assume basic understanding of the concepts of poker and the Hold'em Strategy [67]–[69]. These vignettes are used to evaluate the ability of the entire FAMTILE algorithm, including recognizing the atomic actions of the human.

For this paper, two training vignettes were developed using the Limit Hold'em game. In the first poker vignette (vignette C), only one betting round occurring prior to the *flop* is considered. The human is placed at a random position at a poker table and seated with seven computerized opponents. The dealer button is placed at a random position, and each player is dealt two hole cards. Starting with the player to the left of the big-blind bet, each opponent makes an action (either to fold, call, or raise) until it is the human's turn to act. At this point, the human will know his two hole cards, his position at the table, and the actions of each opponent who has acted before him. The simulation then prompts the human to make an action: either to fold, call, or raise. The human's actions are recorded, along with all applicable observations at that point; then a new hand is dealt, and the player is reseated. This process continues until the simulation has collected a requisite number of observations. A screenshot of the simulation for this vignette is provided in Fig. 8.

For the second poker vignette (vignette D), the human is asked to make decisions throughout entire hands and accumulate chips throughout the vignette. This is depicted in Fig. 9.

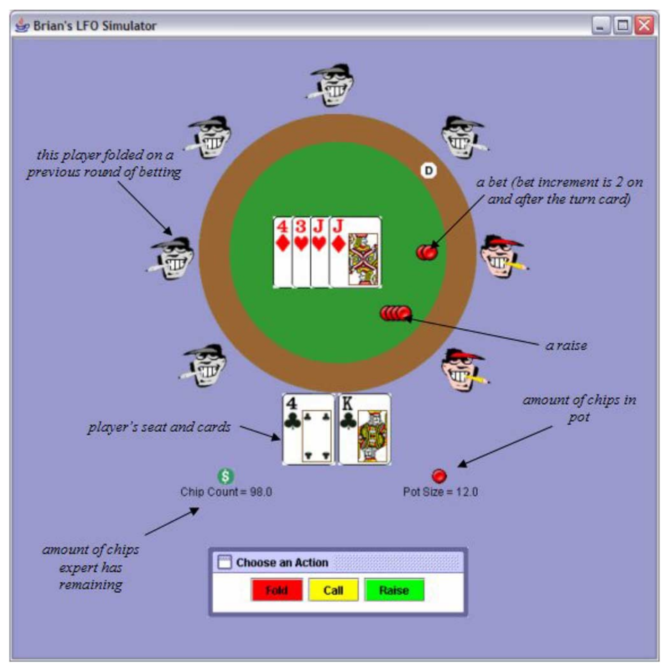

Fig. 9. Vignette D.

This vignette begins just as the first poker vignette—the human is placed at the table with seven opponents, and the button is placed at a random position at the table. A hand is dealt, and each opponent makes an action on their cards until it is the human's turn to act. When the human acts, however, the betting round continues as well as the hand and proceeds just like a standard round of Limit Hold'em. After each round, the dealer button rotates one chair to the left, and a new hand is dealt. A *chip count* is stored for the human, which reflects the amount of money won/lost during the sequence of hands played.

In this vignette, the situations encountered by the human are far more robust and are designed to challenge his playing

TABLE I
RAISE IN POSITION CONTEXT

playerAction = Raise	weight = 6
distanceFromButton = 0	weight = 3
numPlayersInPot = 2	weight = 0.5
numBetsToCall = 1	weight = 0.5

ability. Because the vignette involves entire rounds, the opponents at the table react to the human's decisions and use many of the strategies outlined in [69] to try and win hands. Since this vignette involves the observation of humans playing against opponents, it was important to create opponents who are able to pose at least minimal challenge. Opponents for the vignettes are programmed with the following:

- basic understanding of the strength of its hole cards before the flop;
- basic understanding of the hand strength relative to the cards on the board;
- basic understanding of the hand potential relative to the cards on the board;
- ability to bluff;
- ability to trap or slowplay;
- ability to change play based on position and amount of action in the betting round.

For these vignettes, each action taken by the human must first be interpreted by the TBI engine before presenting a corresponding output pattern to the FAM. This output pattern is the context of the action taken, as interpreted by TBI. Individual actions performed by the human are assumed to be a consequence of the human acting in a particular context. To make an interpretation of the context embodied by the human's recorded action, the TBI engine matched each template against the appropriate conditions present in the observation. The engine then infers the context in which the human is likely to be acting. This determination is then recorded by the interface module and transformed into a bit sequence representing the output pattern for FAM using the technique discussed in the previous section.

In vignettes C and D, we consider a context to be a circumstance and/or rationale for making a particular play. The *raise* action, for instance, is divided into contexts that differentiate the inferred reason for the raise. As discussed by Sklansky [68], there is a variety of purposes behind making a raise: to force weaker hands to fold; to get more money into a pot; to bluff, thereby causing stronger hands to fold; etc. While the human's intent cannot be recorded through strict observation, it can be inferred if each of these purposes is encoded by a context. Using expertise gathered from poker experience and from various texts [67]–[69], a set of contexts that result in each possible action (e.g., raise, call, bet, and fold check) in both vignettes was generated. When an observation is presented to FAMTILE's TBI engine, it is compared against the attributes of each context template and generates a score for that template. Consider the template in Table I for the *RaiseInPosition* context. This context refers to a situation where the human has made a raise based mostly on his strong position relative to the dealer button. As stated earlier, players *on the button* get to act last on each postflop betting round, giving them a significant advantage of being able to react to each opponent's play.

Note the weights associated with each attribute. The most heavily weighted attribute is the player's action: if the player does not make a raise, this weight induces the TBI engine to calculate a low score for this template. The other weights are assigned based on their relevance to the context, i.e.,

$$score_{\text{att}} = \frac{(1 - |att_{\text{observed}} - att_{\text{template}}|)}{range_{\text{att}}} weight.$$

Since the training patterns for the neural network come directly from the observations of the human under study, the amount of diversity among those training patterns is completely dependent on the robustness of the vignette in which that human operates.

Knowledge used for training can only be extracted from observations. Thus, any relevant knowledge not executed within an observed simulation will not be learned by the neural network. Because of this, there will be gaps in the tactical knowledge about situations not encountered by the human during the observation phase. If these gaps are ignored by the learning system, the resultant autonomous agent will have no intelligent response if presented with that unlearned situation. The only defense against these gaps in knowledge is to train the network with as many examples as possible in hopes that they sample as much of the human's knowledge as possible, i.e., provide vignettes in which the human must use all or most of his/her tactical knowledge.

C. *Generating Training Inputs from the Observation*

Generating training points for the maze vignettes is a matter of placing the player and goal at random locations within a fixed maze. Each time the player makes a move, the next training point input pattern becomes either a new random position for both him and the goal (as in vignette A) or the updated maze state based on the direction of the player's previous movement (as in vignette B). The output pattern for that training point is then the action taken by the expert for the corresponding maze state represented by the input pattern. Each of these patterns, however, must first be translated into a readable form, so that they can serve as useful training patterns for FAMTILE. The output pattern is simply the context that the expert has chosen as a response to the stimuli represented by the input pattern.

For the Poker vignettes, the simulation must generate and record the following pieces of information for each observation:

- player's hole cards;
- board cards (vignette D);
- player's position;
- position of the button;
- opponent actions;
- amount of money in the pot (vignette D);
- player's action.

To generate this information, the simulation deals a random hand to the expert and seven automated opponents. Each opponent makes an action until it is the player's turn. At this point, the state of the hand is recorded, along with the action made by the player for his turn. For vignette C, each of these points occurs during the betting round prior to the flop.

For vignette D, this observation is expanded to include interpreted information about the player's hand and position relative

TABLE II
VIGNETTE D CONTEXTS

foldWithWeakHand	Player folds because his cards are weak
foldWithMediocreHand	Player folds an average hand
foldWithDrawingHand	Player folds a good drawing hand
foldWithStrongHand	Player unknowingly folds a strong hand
checkWithWeakHand	Player checks with a weak hand, likely with the intention to fold if there is a bet made
checkWithDrawingHand	Player checks a hand that is on the come to a possible winning hand, and would like to see another card for little to no money
checkWithMediocreHand	Player checks with a marginal hand, likely to observe the action at the table
checkWithMonsterHand	Player checks with a monster hand, to fake weakness and induce action from his opponents
checkWithStrongButVulnerableHand	Player checks with a strong hand that is vulnerable to drawing hands
callWithWeakHand	Player makes an extremely loose call with a weak hand
callWithMediocreHandContext	Player makes a 'loose call' with a hand that 'tighter' players would likely fold. A 'tight' player typically only plays with very strong hands and draws.
callWithDrawingHand	Player calls with good multiway hole cards to see a flop, or if he is on a good draw (to a flush, straight, etc.)
callWithMonsterHand	Player calls with a monster hand, attempting to slow-play his hand
callWithStrongButVulnerableHand	Player calls with a strong hand vulnerable to drawing hands
betWithWeakHand	Player bets with a weak hand to bluff
betWithMediocreHand	Player bets with a marginal hand, either to bluff or to induce a weaker hand to fold
betWithDrawingHand	Player bets a drawing hand on a semi-bluff.
betWithStrongButVulnerableHand	Player bets with a strong hand vulnerable to drawing hands
betWithMonsterHand	Player bets with a nearly unbeatable hand
raiseWithWeakHand	Player makes a raise with a weak hand in order to induce the table to fold (a bluff)
raiseWithMediocreHand	Player makes a raise with a mediocre hand, either to bluff or to induce a weaker drawing hand to fold
raiseWithDrawingHand	Player makes a raise with a strong drawing hand, in an attempt to induce either folds or 'free cards' in later rounds.
raiseWithStrongButVulnerableHand	Player makes a raise with a strong hand that could get drawn out on
raiseWithMonsterHand	Player has a nearly unbeatable hand, and is raising to extract the most amount of chips out of his opponents

to the rest of the table. To do this, the following parameters are used:

- *hole cards:* rank of the player's two hole cards (both are scaled to values < 1);
- *suited*: boolean value indicating whether cards have the same suit;
- *hand strength*: fuzzy value of the player's hand, as calculated in [70];
- *pPot*: fuzzy value representing the potential of the player's hand drawing to a winning hand [70];
- *nPot*: fuzzy value representing the potential of the player's hand decreasing in strength due to future board cards [70];
- *betting round*: 4-bit binary value representing the current betting round;
- *last action*: 4-bit binary value representing what the player did on his last turn to act;
- *pot size*: number of chips currently in the pot, scaled to a fuzzy value < 1;
- *opponent bets in pot*: scaled to a fuzzy value < 1 by the size of the largest bet.

Table II summarizes the contexts used for vignette D. There are a total of 24 contexts. For vignette C, only 12 contexts were used. This is because there are fewer actions available to the player in vignette C (player cannot bet) and, more importantly, the player has less information about his hand (no board cards are shown in vignette C, only preflop action) and therefore cannot classify the situation to the same level of granularity.

When the simulation records the expert's action during the observation, the result is simply a character value representing either a raise, fold, or call. For both poker vignettes, however, FAM is used to create a mapping between the observed situation and the expert's choice of context, and not simply his action. To make this transformation, the interface extracts necessary variables from the input pattern to present to the TBI engine, which makes a prediction of the most likely context that the expert has chosen. For vignette C, there are 12 contexts from which the expert can select.

An output pattern for vignette C would therefore be a 12-bit binary number with all but one number set to zero. That number, in the jth position, represents that the TBI engine has identified context j as the active context for the observation represented by the input pattern.

In this prototype, the FAM clusters are stored as 1-D arrays—one for each cluster in the ART$_a$ and ART$_b$ modules. Each entry in these arrays represents a field value of that cluster. To store the mappings, a separate array is created that represents the InnerART module of the FAM. This array contains one field for each cluster created in ART$_a$. The value stored in each field is the index of its mapped cluster in ART$_b$. For instance, if the ART$_a$ cluster i is mapped to cluster j in ART$_b$, the InnerART array would look like $[ia_1, ia_2, \ldots, ia_c = j, \ldots]$. Here, the field containing the value j is stored in the ith slot.

D. Comments on the Application Selected

Two issues that demand some discussion and further explanation come to mind. We address these in this section.

The data obtained were observed from a simulation of games, rather than from watching humans play the game in the real world. This is particularly true for the poker-based vignettes (C and D). The nature of vignettes A and B is such that they really must be played in a computer for them to make much sense. The reason for using a simulation, of course, was to maintain control on the data and avoid noise from the environment. Given that proof of concept of the learning of transitions was the main objective of this paper, we believe that this is justified. However, the question on how one would apply this approach when observing an actor in the real world arises. Our response is that, in an ideal world, our approach could be used in such a situation as long as the features of the actor's actions could be extracted from the observations logged by some front-end process. For example, in poker, the motion of throwing down the card played signals a player's move. The front-end process would have to interpret this move and then focus on the card played to identify it. Alternatively, folding is signaled by laying down all cards and pushing them away from the player. Once this information is fed to our learning system, it would see no difference from having observed a simulation. However, the envisioned front-end process would be quite complex and beyond the scope of this research, at least for the poker application.

More generally, the feasibility of building an adequate front-end process to extract the features would depend on the application, i.e., the type of task being observed and learned. In the case of a physical task or activity where only the location, direction, and speed of a person or a vehicle become important, then a Global Positioning System transmitter that identifies these data to an observer may be sufficient to learn that actor's or vehicle's behavior. This was shown by Fernlund *et al.* [5], albeit using a different approach to learning from observation. However, applications that heavily depend on gestures or human gesticulated motions (such as throwing down cards) may require highly complex front ends to permit their use in learning from observation and would thereby be more limited in their application.

The second question that arises is whether this approach would work in continuous games or tasks that are not turn based. Clearly, turn-based games provide a natural cue for the context to potentially change. Such would not be the case in many continuous tasks such as controlling a vehicle (e.g., car and aircraft). While knowing the time of this (potential) context transition cue clearly simplifies the learning, we can project how such a system would work.

Our approach would be to look for an "interesting" action or event in the sequence of events being observed. Such an "interesting" event would indicate the triggers for the change in context, which is what we are trying to learn. The change in context itself could be identified by a TBI engine by identifying when a new template is used to describe the actions of the human actor. "Interesting" activities would include events, changes in behavior (e.g., slowing down and changing direction), the actions of others (e.g., an enemy fires upon the human actor), environmental occurrences (e.g., it starts to rain), or even geographical location (e.g., passing a landmark and reaching an exit in an interstate highway). Of course, the crux of this approach would be carefully defining the concept of "interesting," as well as determining how to identify all such events and actions just before and after the transition. Events and actions after the transition takes place could indicate anticipation by the human actor. While we did not address the issue of temporally continuous actions, it does remain an interesting subject of future research.

V. Testing and Evaluation of Concept

We subjected the prototype FAMTILE system to six *test scenarios* (TSs) to determine whether the concepts behind the prototype—the use of neural networks to learn context transition criteria from observation of a human performer—work as expected. As described in the previous section, we have developed four *vignettes* (A–D), each presenting the human test subjects with a different game in which to make decisions. We designed the six TSs to evaluate the effectiveness of our work. TSs 1 and 2 involve the first two vignettes, whereas TSs 3, 4, 5, and 6 involve the poker vignettes (C and D).

For this evaluation, four human test subjects (denoted here as *Alpha*, *Bravo*, *Charlie*, and *Delta*) are used. Three subjects participated in each of the four vignettes, but they were different ones for the various vignettes. This was done to accommodate their varying availabilities. The subjects were selected from a pool of students in the laboratory that had some experience with poker. Three of the subjects were male (Alpha, Bravo, and Charlie), whereas subject Delta was a female. With regard to the poker vignettes, two of the three participating subjects (Alpha and Charlie) considered themselves to be of moderate to advanced skill, whereas subject Delta was a relative novice. These subjects were asked to install the vignettes on their computer and play the games while the simulation recorded each of their decision points.

Subjects Alpha, Bravo, and Delta participated in TSs 1 and 2. These scenarios correspond to vignettes A and B, respectively, and evaluate the ability of FAM to learn relatively simple behaviors exhibited by the test subject in these vignettes, without the TBI context identification feature. The basic objective of TSs 1 and 2 was to evaluate the ability of a standalone FAM to learn human-performed actions in a simple game before applying them to a more complex game. In TSs 1 and 2, atomic actions are represented by directional choices: either left, right, up, or down. These directions are also representative of the entire action space of the behavior, as no other actions are permitted within the maze. In vignettes A and B, all possible contexts that may provide motivation for each action are ignored during training. For instance, the motivation of going left because the goal state is in that direction is considered to be identical to the motivation of going left simply because that is the best alternative. Because of this, contexts behind the selection of particular moves by the test subjects were not considered in these two testing scenarios. We should note, however, that contexts still exist on the part of the agent that moves in the simulation. It is just that they are not considered in the training.

In TSs 3 and 4, subjects Alpha, Charlie, and Delta performed the more complex activities related to vignettes C and D, respectively: participating in hands of Texas Hold'em. The objective of TSs 3 and 4 was to evaluate the ability of a standalone FAM system to learn the actions and play them back in a simulated game, regardless of the underlying contexts. The learning poker agent merely learned to map the game conditions (the environment) to the actions taken by the test subjects. Comparison of the results of TSs 3 and 4 later to those of TSs 5 and 6 would, furthermore, provide an indication of the value of learning to predict the underlying contexts rather than merely the actions. Vignettes C and D involve reasoning about several observations, where each may have a significant impact on the subject's eventual decision. Furthermore, each action taken by the subject may be the result of complex motivations, as would be appropriately defined in a context. For instance, a raise or a bet resulting from the action prescribed in one context may be caused by a different reason than it would in another context. TSs 3 and 4, however, intentionally ignore this fact. When a player makes an action, it is presented to FAM as that action, regardless of any context that may be behind it. Because of this, these tests mirror those of TSs 1 and 2, but with significantly more complex behaviors.

TSs 5 and 6 also employ vignettes C and D, respectively, and were executed by subjects Alpha, Charlie, and Delta. By contrast, TSs 5 and 6 consider the context of each subject action prior to creating a training pattern for the neural network. Before running TSs 5 and 6, a set of contexts was developed for both vignettes C and D in an effort to capture all possible motivations for each action. During training, the subject's action at each decision point is first examined by a TBI engine to infer a context for that point. In TS 5, vignette C is reused, and FAMTILE attempts to learn subject actions just as FAM attempted to do in TS 3. It is hypothesized that the representation of the subjects' actions as inferred contexts can help the network to more effectively make finer clusters representing more closely related patterns, thereby increasing the predictive accuracy of the system. For the FAM within FAMTILE, just as in TSs 3 and 4, the actions of the observed human performer (the subject) are presented as output patterns, regardless of the motivation behind the action.

A. Evaluation Procedure

The evaluation of the FAM learning process for TSs 3 and 4 was done as presented here.

- The entire observation sequence gathered from subject i is used to generate a set of training patterns—no validation set is generated.
- FAM is trained with a set of patterns and learns a mapping between observation and action.
- FAM replaces the test subject and is presented with various decision points as the game progresses.
- For each decision cue presented by the simulation, FAM predicts an action based on what it learned.
- That action is then executed in the simulation, and the vignette continues.
- The overall performance of both subject i and FAM is compared based on the metrics collected throughout the execution of the scenario.

When separately testing FAM (TSs 3 and 4), the network is trained with the subject's action being presented at its output. For FAMTILE (TSs 5 and 6), the actions of the subject are first translated to an inferred context (by the TBI) for each decision point, and a representation of that context is presented to the FAM network within FAMTILE. After the training of each system was completed, the simulation was run again. This time, each decision cue was presented to the newly trained poker agent. Based on its knowledge, then, the poker agent running FAMTILE predicts a context, and the actions associated with that context were executed. In contrast, the standalone FAM produces only a predicted action. Six steps for testing the full FAMTILE system are given here.

1) The entire observation sequence gathered from subject i is used to generate a set of training patterns. Both the training and validation sets are taken from these observations.
2) FAMTILE is trained with the complete set of patterns and generates a mapping between the observation and the context.
3) FAMTILE takes the place of the subject within the simulation and executes the appropriate vignette.
4) For each decision cue presented by the simulation, FAMTILE predicts a context.
5) The identified context provides an appropriate action that is then executed. The vignette continues.
6) The overall behaviors of both subject i and FAMTILE are compared based on the metrics collected throughout the execution of the vignette.

For each scenario, the following FAM parameters were held constant:

- $\varepsilon = 0.00001$;
- $\beta_a = \beta_b = 1$;
- $\rho_b = 1$.

TABLE III
SUMMARIZED RESULTS FOR SCENARIO 1

Subject	$\overline{\rho}_a$	$\overline{\rho}_{a_{test}}$	$\overline{\mu}$	$\overline{\sigma}$
Alpha	0.6	0	94.7	2.38
Bravo	0.8	0	87.3	3.27
Delta	0.8	0	80.6	3.76

The only parameter that was modified during the testing phase was the baseline vigilance $\overline{\rho}_a$. This parameter has a direct effect on the granularity of the clusters generated in the ART_a module. These clusters represent groups of input patterns presented to ART_a, where each pattern maps to the same output pattern (either an action as in TSs 1, 2, 3 and 4, or a context as in TSs 5 and 6) and is closely matched with respect to its individual field. The baseline vigilance parameter $\overline{\rho}_a$ affects this granularity by raising the vigilance parameter, which is responsible for rejecting the addition of new input patterns to a certain cluster if it fails to match a certain criteria. This change ultimately increases the number of input pattern clusters created in ART_a by decreasing their overall size (and inclusiveness). This effect is quantitatively illustrated in the succeeding sections.

B. TS 1 Results

Essentially, the task for FAM in this TS is to create a mapping between the maze topology and a predicted direction for the test subject facing that situation: either left, right, up, or down.

The intent of vignette A is to create an environment where the actions of the subject are closely tied to the primary goals of the behavior. In this vignette, the subject makes only a single move in response to being told where and how far away the goal position is. Each atomic move, therefore, is made in direct accordance with the objective of reaching the goal. In the next few vignettes, the behavior required becomes increasingly complex, and the relationship between the atomic actions required by the subject consequently become less dependent on the overall objective and more dependent on the context in which the subject is operating.

The testing proceeded in five steps.

1) Randomize the order of the 1000 training points.
2) Choose 900 of the 1000 points at random to train the neural network; use the final 100 points for the validation set.
3) Train the neural network using the 900 chosen training points.
4) Test the neural network using the remaining 100 points.
5) Record the number of correct predictions made out of 100 testing patterns.

Table III displays the results for each subject, including the sample mean predictive accuracy μ and standard deviation $\overline{\sigma}$. A 2-tailed t-test was used on each set of data to validate that the computed sample mean $\overline{\mu}$ for each subject approaches the actual mean μ. Using an α value of 0.01, the test computed a 99% confidence interval for the actual mean.

As expected, FAM is able to successfully learn the movement patterns for each of the three subjects. Success, here, is defined as better than random. A random guess at the subject's action for vignette A would yield, on average, 25% predictive accuracy (because there are four possible actions). As a qualitative

TABLE IV
SUMMARIZED RESULTS FOR TS 2

	$\overline{\rho}_a$	$\overline{\rho}_{a_{test}}$	$\overline{\mu}$	$\overline{\sigma}$
Alpha	0.8	0	92.5	2.63
Bravo	0.8	0	84.5	3.42
Delta	0.7	0	85.6	3.31

TABLE V
AVERAGE PREDICTIVE ACCURACY FOR TS 3 USING OPTIMAL $\overline{\rho}_a$ VALUES

	$\overline{\mu}$	$\overline{\sigma}$
Alpha	75.04	4.20
Delta	68.54	4.46
Charlie	75.56	3.68

comparison, consider the accuracies achieved by each subject. For subject Alpha, the network was able to predict, on average, almost 95 of the 100 testing patterns. Even for the worst cased subject (TS 3), FAM was able to predict nearly 81% of the testing patterns.

The purpose is for these results to serve as a baseline to evaluate FAM (and ultimately FAMTILE) and examine how this notion of context affects their predictive accuracy.

C. TS 2 Results

TS 2 was executed in the same manner as TS 1, and the same three subjects were used. Within this scenario, each subject makes consecutive moves within a 10 $\times$ 10 maze, with the board and goal positions resetting each time the subject reaches the goal. The scenario ends when the subject has generated 1000 training points—each training point represents a specific maze state and the action the subject makes in response to that state. Those points were used to train and evaluate the neural network. Table IV displays the results of the 1000 run sets for each subject.

In this scenario, FAM was able to adequately learn the movement patterns for each of the three subjects. Furthermore, the predictive accuracy significantly varied across subjects, just as it had in scenario 1. FAM achieved a predictive accuracy of nearly 93 of 100 for subject Alpha versus 84.5 and 85.6 for the other two.

D. TS 3 Results

In vignette C, each of three test subjects is placed at a simulated Texas Hold'em game with seven computer-generated opponents. As expected, the predictive accuracy of FAM significantly degraded when tested using vignette C as a result of the greater complexity of the problem. By the numbers, the network achieved best-case predictive accuracies of 75.0, 68.5, and 75.6 for each player versus 92.5, 84.5, and 85.6 for TS 2, respectively (see Table V).

Comparing the predictive accuracies of FAM on these two subjects for TSs 2 and 3, there is a 17.5% decrease in predictive accuracy for subject Alpha and a 17.1% decrease for subject Delta. This is a sharp contrast to the statistically insignificant performance difference between TSs 2 and 1, where the network's predictive accuracy changed to 2.2% and 2.8% for subjects Alpha and Delta, respectively. These results confirm that the poker environment of vignette C is much more complex and therefore harder for FAM to learn versus that of the simpler maze vignettes. What this means in terms of the network itself is that FAM had a more difficult time effectively creating clusters with similar data points that mapped to the output patterns representing correct predictions of the subject's action.

An interesting result of this test was the sharp contrast in the predictive accuracy of FAM for subject Delta versus the other two subjects. As previously reported, FAM was only able to predict 68.54% of subject Delta's actions versus 75.04 and 75.56% for the other two subjects. One hypothesis as to this discrepancy is the difference in skill between subject Delta and subjects Alpha and Charlie. In Texas Hold'em, proper play before the flop is both the easiest piece of strategy to learn and the most crucial [69]. Strategy after this round becomes much more complex because of the explosion of information present with community cards on the board. Because of this, Limit Hold'em play before the flop round of betting tends to be somewhat mechanical among experienced players. This is supported by the data on subjects Alpha and Charlie, who shared similar experiences and read much of the same literature. Subject Delta (the novice player as previously described), on the other hand, has much less experience; thus, her play is likely to be more erratic and, therefore, less predictable. However, a similar drop-off between subject Delta versus subjects Alpha and Charlie is present in the results reported in scenario 1 (although not in scenario 2). Because of this, another hypothesis for the change in predictive accuracies is the level of attention Delta paid to the exercise for vignettes A and C. Since the participants did not execute each vignette in sequence (and was not monitored during the exercises), it is possible that Delta simply was not paying full attention during the exercises. This hypothesis is bolstered by the more reasonable results of scenario 2, where the decision points were much more straightforward (navigating an entire maze versus simply making a single decision of direction).

E. TS 4 Results

In TS 4, the predictive accuracies for FAM were collected and analyzed for vignette D. Just as vignette C, this vignette is set at the poker table with seven computer-generated agents playing against the subject in games of Texas Hold'em. Here, however, the subject's decision points are not limited to the first round of action. Instead, a series of entire hands are carried out to their completion: if a subject folds, a new hand is dealt; if a subject raises, the opponents accordingly react to that raise; a flop, turn, and river are dealt; and betting rounds follow just as in an actual hand. The subject is also given a stack of 100 "chips" that is maintained throughout the vignette. In this fourth and final evaluation of the FAM, we continue to examine its ability to learn subject actions as a function of his cards, his position at the table, and the betting action.

Once again, the increase in complexity of vignette D compared to vignette C resulted in further erosion in the FAM's predictive accuracy. The best-case accuracies of 55.32, 58.95, and 58.12 (see Table VI) are an average of more than 20% worse than those of scenario 3, which is nearly twice the decrease observed between vignette C and the maze scenarios.

TABLE VI
AVERAGE PREDICTIVE ACCURACY FOR TS 4 USING OPTIMAL $\overline{\rho}_a$ VALUES

	$\overline{\mu}$	$\overline{\sigma}$
Alpha	55.32	5.24
Charlie	58.95	4.47
Delta	58.12	2.91

TABLE VII
SUMMARIZED RESULTS FOR SCENARIOS 3 AND 5

	$\overline{\mu}_1$	$\overline{\mu}_2$	$\overline{\mu}_1-\overline{\mu}_2$	**99%CI**	**p-value**
Alpha	75.63	75.40	0.224	(-0.228,0.676)	0.201
Delta	68.92	68.55	0.372	(-0.135,0.879)	0.059
Charlie	75.37	75.56	-0.187	(-0.666,0.292)	0.315

It was observed in TS 3 that FAM significantly performed worse on Delta than on the other two experts. Furthermore, it was noted that Delta had several years fewer experience than the other two, which possibly affected the predictability and consistency of the actions.

The complexity of this scenario, however, seems to have neutralized this effect. In fact, FAM was slightly more effective in the best case at predicting expert Delta's actions than those of the other two experts. As it turns out, Charlie (who did not participate in vignette C or the maze vignettes) had comparable experience as expert Alpha.

F. TS 5 Results

The objective for TS 5 is to evaluate FAMTILE's ability to predict both the subject's inferred active context and his resultant action. Vignette C is used for this TS, which is the same one used to evaluate FAM in testing scenario 3. Because of this, the results of TS 3 serve as a baseline performance metric for the results achieved here. Unlike FAM, however, FAMTILE instead attempts to predict the subject's inferred active context. In order to make a comparison between FAM and FAMTILE, the predicted contexts of FAMTILE must then be converted to a predicted action for the subject, using the contents of the predefined context template. Because FAM does not make context predictions, this determination is necessary to compare the predictive accuracies of the two learning systems. The results of scenario 5 are presented in Table VII (represented by $\overline{\mu}_1$), along with those from scenario 3 (represented by $\overline{\mu}_2$), using 900 training patterns.

There are several interesting things to note from these results. In terms of the primary objectives of this research, the numbers in the third column are the most important—how well does FAMTILE predict the inferred context of the subject? As Table VII illustrates, these predictive accuracies of the subject's action for FAM and FAMTILE are nearly identical for each batch of runs and each subject. In the best case, for subject Alpha with 900 training patterns, FAMTILE outperformed FAM with an average of 75.63 correct predictions versus 75.04 for FAM. In the worst case, for subject Delta, FAM narrowly outperformed FAMTILE with an average of 75.56 correct predictions versus 75.37 for FAMTILE. However, neither of these margins is statistically significant.

TABLE VIII
AVERAGE CONTEXT-PREDICTIVE ACCURACY FOR TS 5

	$\overline{\mu}$ **(context)**	$\overline{\sigma}$
Alpha	67.71	4.04
Delta	59.98	4.81
Charlie	66.26	5.17

TABLE IX
SUMMARIZED RESULTS FOR SCENARIOS 4 AND 6

	$\overline{\mu}_1$	$\overline{\mu}_2$	$\overline{\mu}_1-\overline{\mu}_2$	**99%CI**	**p-value**
Alpha	60.25 (75.63)	58.22 (75.40)	2.30	(1.253,3.347)	0.778
Delta	60.14 (68.92)	60.18 (68.55)	-0.04	(-0.460,0.380)	0.006
Charlie	54.07 (75.37)	55.32 (75.56)	-1.25	(-2.38,-0.120)	0.572

In addition, FAMTILE is able to accurately predict the subject's active context an average of 67.71, 59.98, and 66.26 times for each of the three subjects observed, respectively, at optimum values for $\overline{\rho}_a$ (see Table VIII). Comparing these accuracies with those of FAM for predicting subject actions, we note that FAMTILE is an average of only 11.52% less effective at predicting contexts than FAM is at predicting actions.

The fact that FAMTILE is able to generate a competitive degree of context-predicting accuracy *without* disrupting the ability of FAM is significant. In effect, therefore, we have created a system that adds the ability to predict context transitions to a neural network without significantly affecting its ability to predict simple actions.

G. TS 6 Results

In scenario 6, predictive accuracies for FAMTILE are collected and analyzed for vignette D as they were for FAM in scenario 4. Table IX summarizes the results of a 2-tailed t-test on the best-case predictive accuracy means achieved in scenarios 4 ($\overline{\mu}_2$) and 6 ($\overline{\mu}_1$) for each subject. In the table, the values from scenarios 3 and 5 are annotated in parentheses.

The predictive accuracy of FAMTILE for predicting the subject's inferred context also considerably decreased from the values achieved in scenario 5. Whereas FAMTILE predicted contexts at rates of 67.71, 59.98, and 66.26 for vignette C, those accuracies dropped by an average of more than 28% across the two subjects who then also participated in vignette D. One significant reason for this was the increase in the number of contexts. This number doubled from 12 to 24 contexts for vignette D, because two new actions needed to be accounted for (i.e., bet and check), along with the representation of contexts potentially present after the preflop round of betting. Note that, with 24 contexts, a random guess of the inferred active context could be expected to be correct slightly more than 4% of the time, which is ten times less than the accuracy achieved by FAMTILE.

Furthermore, vignette D requires the player to reason about entirely new and more complex situations than those faced in vignette C. In addition to his/her hole cards, the player must also consider not only the community cards but also the action of previous betting rounds and the possible responses of each opponent in response to a particular action.

VI. Conclusion and Lessons Learned

Based on the results tabulated in the previous section, it is concluded that FAMTILE is an adequate technique for learning high-level behaviors and offers several promising characteristics that can be exploited in future research. Because it is able to learn low-level contexts from human actors without adversely affecting the clustering ability of FAM, we feel that the FAMTILE system provides a significant tool for learning in systems where it is desirable to gain a perspective of *why* the human actor is doing what he/she is doing.

The results of the two maze scenarios provide a good indication as to FAM's ability to predict human responses to an observation. In TS 1, the network is able to correctly predict a subject's movement at an average of 86% on the validation set, achieving nearly a 95% average for one of the three subjects. This scenario included input training patterns with 27 fields and four possible output patterns. The second maze TS expanded the subject's viewing range, more than tripling the number of input-pattern fields to 88 (92 if the subject's previous action was recorded and considered). Nevertheless, FAM is able to predict 85% of the validation set for the three subjects, increasing to nearly 87% when the subject's previous action is considered.

While these are impressive numbers for predicting three different subject's actions, they only speak to the successes of FAM and do not address the capabilities of FAMTILE. These scenarios were executed and reported, for the most part, to justify the use of FAM for doing the low-level learning task. Had these evaluations been a failure, a different learning system would have had to be selected—one that performed better at predicting actions within these training scenarios.

As described in Sections IV and V, FAMTILE requires the use of a completely separate TBI module that encodes *a priori* knowledge about the scenario within its context templates, whereas FAM itself requires no such input. FAMTILE fails to produce a worthwhile increase in predictive performance, therefore negating our hypothesis. A separate set of tests was run to evaluate FAMTILE's ability to correctly predict the inferred expert context for each decision point. While these tests resulted in lower predictive accuracies—certainly expected because the neural network must choose between 12 possible output patterns, instead of only three, when predicting actions—the results were promising. Using 900 training patterns, FAMTILE is able to correctly predict an average of 64.77 contexts out of a possible 100 (64.77%) across the three experts. As reported in Section VI, FAMTILE's predictive accuracy for contexts is only around 11% worse than its accuracy for actions. This accuracy is achieved, furthermore, without affecting the accuracy of the network in predicting the expert's overall action. What this means, then, is that FAMTILE can provide a significant advantage over other supervised learning algorithms in situations where the identification of expert context provides more important or additionally worthwhile information versus simply being able to predict low-level action. In a more robust poker simulation, for example, the ability of FAMTILE to identify context could drive additional behaviors, aside from the simple game action, such as additional "table talk" to project a strong image while bluffing, voice intonation, etc. Generally, we feel that the FAMTILE system is most useful for learning tasks where three conditions hold.

1) The behavior satisfies the characteristics of high-level tactical behavior, as defined in Section I.
2) The user is interested in creating models of the expert's behavior and is more interested in his resultant intentions and motivations than the actions observed at the lowest level.
3) The expert's ultimate action is more closely tied to his low-level behavior than to the raw observation presented at each decision point.

This difference in difficulty between the maze and the poker vignettes seemed to create a good set of conditions for evaluating both FAM and FAMTILE. The first human-prediction task (the maze) was found to be relatively easy yet reflected some variability among the three subjects observed. The second two TSs introduced the poker scenario. These vignettes introduce a learning challenge that, while containing a comparable number of input-pattern fields and output possibilities, proved to be a more difficult task for both FAM and FAMTILE.

FAMTILE requires the use of a separate TBI module that encodes *a priori* knowledge about the scenario within its context templates, whereas FAM itself requires no such input. FAMTILE fails to produce a worthwhile increase in predictive performance.

The central assumption made for this research was that high-level behavior can be represented by a sequence of lower level behaviors that can be modeled by CxBR contexts. However, the trick then becomes defining and partitioning each context of a behavior in such a manner that they are truly atomic and identifiable, independent of the specific subject being observed. For example, consider the *RaiseWithStrongButVulnerableHand* context used in vignette D. This context was modeled to represent cases where the subject believes not only that he has the best hand at the moment but also that his opponents can easily draw cards to beat him.

This context raises an interesting question: What if the subject does not actually recognize this? Obviously, then, the templates must be defined such that this context is not inferred. However, what if there are no contexts that accurately represent the low-level motivation and behavior of the human subject?

High-level behaviors whose specifics are heavily dependent on human preference and expertise are equally difficult to represent. While a significant amount of *a priori* knowledge was encoded into the context templates used for scenarios 3 and 4, that knowledge does not represent the full range of motivations and contexts that constitute the entire task of playing Hold'em Poker. This is because these contexts are so dependent on the tendencies of the individual subject. Some players may employ poor strategies, for instance, that are not represented as a high-level context template. These absences can ultimately reduce the predictive accuracy of the FAMTILE system.

However, that is not to say that these assumptions serve only to doom the chances of success for our approach. On the contrary, these assumptions provide a means for motivating the directions that research in human behavior representation should progress. If we choose to learn a task where the modeling architecture, subject tendencies, and context topologies are all known, it is likely that the task modeled is too simple and not worth modeling. Texas Hold'em Poker, on the other hand, is a highly complex game, and the number of techniques, strategies, and styles documented and used by advanced players suggest that the game is as much of an art as it is a science.

REFERENCES

[1] *Webster's New World Dictionary of the American Language*, 1960, Cleveland, OH: World.

[2] A. J. Gonzalez and R. Ahlers, "Context-based representation of intelligent behavior in training simulations," *Trans. Soc. Comput. Simul. Int.*, vol. 15, no. 4, pp. 153–166, Mar. 1998.

[3] A. J. Gonzalez, J. Castro, and W. J. Gerber, "Automating the acquisition of tactical knowledge for military missions," *J. Defense Model. Simul.*, vol. 3, no. 1, pp. 145–160, Spring 2006.

[4] T. A. Sidani and A. J. Gonzalez, "A framework for learning implicit expert knowledge through observation," *Trans. Soc. Comput. Simul. Int.*, vol. 17, pp. 54–72, 2000.

[5] H. Fernlund, A. J. Gonzalez, M. Georgiopoulos, and R. F. DeMara, "Learning tactical human behavior through observation of human performance," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 128–140, Feb. 2006.

[6] R. M. Jones, J. E. Laird, P. E. Nielsen, K. J. Coulter, P. G. Kenny, and F. Koss, "Automated intelligent pilots for combat flight simulation," *AI Mag.*, vol. 20, no. 1, pp. 27–41, 1999.

[7] R. M. Jones and J. E. Laird, "Constraints on the design of a high-level model of cognition," Univ. Michigan, Ann Arbor, MI, 1996. Res. Rep.

[8] D. J. Patterson, L. Liao, D. Fox, and H. Kautz, "Inferring high-level behavior from low-level sensors," Univ. Washington, Seattle, WA, 2003. Res. Rep.

[9] R. Khardon, "Learning to take actions," Dept. Comput. Sci., Univ. Edinburgh, Edinburgh, U.K., Jun. 12, 1998. Tech. Rep.

[10] M. van Lent and J. E. Laird, "Learning hierarchical performance knowledge by observation," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 229–238.

[11] C. E. Shannon, "Programming a computer for playing chess," *Philos. Mag.*, vol. 41, no. 314, pp. 256–275, 1950.

[12] A. M. Turing, "Digital computers applied to games," in *Faster Than Thought: A Symposium on Digital Computing Machines*, B. V. Bowden, Ed. London, U.K.: Pitman, 1953, ch. 25, pp. 286–310.

[13] A. Newell, J. C. Shaw, and H. A. Simon, "Chess-playing programs and the problem of complexity," *IBM J. Res. Develop.*, vol. 2, no. 4, pp. 320–335, 1958.

[14] N. Charness, "Components of skill in bridge," *Can. J. Psychol.*, vol. 33, no. 1, pp. 1–16, Mar. 1979.

[15] N. Charness, "Component processes in bridge bidding and novel problem-solving tasks," *Can. J. Psychol.*, vol. 41, no. 2, pp. 223–243, Jun. 1987.

[16] N. Charness, "The impact of chess research on cognitive science," *Psychol. Res.*, vol. 54, no. 1, pp. 4–9, Mar. 1992.

[17] T. Jastrzembski, N. Charness, and C. Vasyukova, "Expertise and age effects on knowledge activation in chess," *Psychol. Aging*, vol. 21, no. 2, pp. 401–405, Jun. 2006.

[18] F.-H. Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton, NJ: Princeton Univ. Press, 2002.

[19] M. Newborn and D. Kopec, "The twentieth annual ACM North America computer chess championship," *Commun. ACM*, vol. 33, no. 7, pp. 92–105, Jul. 1990.

[20] D. Coursey, "Deep Thought is a chess-playing computer," *MIS Week*, vol. 11, no. 14, p. 33, Apr. 2, 1990.

[21] A. L. Zobrist, "Feature extraction and representation for pattern recognition and the game of GO," Ph.D. dissertation, Univ. Wisconsin, Madison, WI, 1970.

[22] J. L. Ryder, "Heuristic analysis of large trees as generated in the game of Go," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1971.

[23] A. Kierulf and J. Nievergelt, "Computer GO: A smart board and its applications," in *Go Worlds*, vol. 42. Tokyo, Japan: Ishi Press, Winter 1986.

[24] A. Kierulf, K. Chen, and J. Nievergelt, "Smart game board and go explorer," *Commun. ACM*, vol. 33, no. 2, pp. 152–167, Feb. 1990.

[25] B. Wilcox, "Reflections on building two GO programs," *SIGART Newsl.*, vol. 94, pp. 29–43, Oct. 1985.

[26] A. L. Samuel, "Some studies in machine learning using the game of checkers," in *Computers and Thought*, E. A. Feigenbaum and J. Feldman, Eds. New York: McGraw-Hill, pp. 71–105.

[27] D. A. Waterman, "Generalization learning techniques for automating the learning of heuristics," *Artif. Intell.*, vol. 1, no. 1/2, pp. 121–170, 1970.

[28] R. S. Michalski, J. G. Carbonel, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*. Los Altos, CA: Morgan Kaufmann, 1986.

[29] M. X. Zhou and S. Ma, "Toward applying machine learning to design rule acquisition for automated graphics generation," IBM T. J. Watson Res. Center, Hawthorne, NY, 2000. Res. Rep.

[30] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to fly," in *Proc. 9th Int. Conf. Mach. Learn.*, Aberdeen, U.K., 1992, pp. 335–339.

[31] R. Camacho, "Inducing models of human control skills," in *Proc. 10th ECML*, Chemnitz, Germany, 1998, pp. 107–118.

[32] A. Isaac and C. Sammut, "Goal-directed learning to fly," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, DC, 2003, pp. 258–265.

[33] A. E. Henninger, "Neural network based movement models to improve the predictive utility of entity state synchronization methods for distributed simulations," Ph.D. dissertation, School Electr. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, 2001.

[34] W. J. Gerber, "Real-time synchronization of behavioral models with human performance in a simulation," Ph.D. dissertation, School Electr. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, 2001.

[35] M. J. Johnson, M. McGinnis, M. Mollaghasemi, and T. Damarly, "Methodology for human decision making using fuzzy ARTMAP neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Honolulu, HI, 2002, pp. 2668–2673.

[36] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sep. 1992.

[37] M. van Lent and J. E. Laird, "Learning procedural knowledge through observation," in *Proc. 1st Int. Conf. K-CAP*, Victoria, BC, Canada, 2001, pp. 179–186.

[38] T. Konik and J. E. Laird, "Learning goal hierarchies from structured observations and expert annotations," in *Proc. 14th Int. Conf. Inductive Logic Program.*, 2004, pp. 198–215.

[39] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, Jun. 1999.

[40] S. Walczak and P. Fishwick, "A quantitative analysis of pattern production and its relationship to expert performance," *J. Exp. Theor. Artif. Intell.*, vol. 9, no. 1, pp. 83–101, Jan. 1997.

[41] J. E. Laird, P. Rosenbloom, and A. Newell, *Universal Subgoaling and Chunking*. Norwell, MA: Kluwer, 1986.

[42] D. Pomerlau, C. Thorpe, D. Longer, J. K. Rosenblatt, and R. Sukthankar, "AVCS research at Carnegie Mellon University," in *Proc. Intell. Vehicle Highway Syst., Amer. Annu. Meeting*, 1994, pp. 257–262.

[43] D. Bentivegna and C. Atkeson, "Learning from observation using primitives," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, May 21–26, 2001, pp. 1988–1993.

[44] A. Moukas and G. Hayes, "Synthetic robotic language acquisition by observation," in *Proc. 4th Int. Conf. Simul. Adapt. Behav.*, 1996, pp. 568–579.

[45] B. Yang and H. Asada, "Hybrid linguistic/numeric control of deburring robots based on human skills," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1992, vol. 2, pp. 1467–1474.

[46] D. Floreano and F. Mondada, "Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot," in *Proc. Conf. Simul. Adapt. Behav.*, 1994, pp. 421–430.

[47] A. Pentland and A. Liu, "Toward augmented control systems," in *Proc. Intell. Vehicles*, 1995, vol. 1, pp. 350–355.

[48] L. J. Fogel, W. Porto, and M. Owen, "An intelligently interactive non-rule-based computer generated force," in *Proc. 6th Conf. Comput. Generated Forces Behavior Representation*, Orlando, FL, Jul. 23–25, 1996, pp. 265–271.

[49] J. D. Morrison, "Real time learning of doctrine and tactics using neural networks and combat simulations," *Mil. Oper. Res.*, vol. 2, no. 3, pp. 45–60, 1996.

[50] M. Crowe, "The application of artificial neural systems to the training of air combat decision-making skills," in *Proc. 12th Inter-Service/Industry Training Syst. Conf.*, Orlando, FL, 1990, pp. 302–312.

[51] H. Friedrich, M. Kaiser, and R. Dillman, "What can robots learn from humans?," *Annu. Rev. Control*, vol. 20, pp. 167–172, 1996.

[52] M. Kaiser and R. Dillman, "Building elementary robot skills from human demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, 1996, pp. 2700–2705.

[53] S. Rajput, C. R. Karr, J. E. Cisneros, and R. J. Parsons, "Learning the selection of reactive behaviors," in *Proc. 6th Conf. Comput. Generated Forces Behavioral Representation*, Orlando, FL, Jul. 23–25, 1996, pp. 3–8.

[54] M. R. Hieb, G. Tecuci, and M. Pullen, "Training ModSAF command agent through demonstration," in *Proc. 6th Conf. Comput. Generated Forces Behavioral Representation*, Orlando, FL, Jun. 1996.

[55] C. G. Gingrich, D. R. Kuespert, and T. J. McAvoy, "Modeling human operators using neural networks," *ISA Trans.*, vol. 31, no. 3, pp. 81–90, 1992.

[56] G. E. Hovland, P. Sikka, and B. J. McCarragher, "Skill acquisition from human demonstration using a hidden Markov model," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1996, pp. 2706–2711.

[57] K. Kosuge, T. Fukuda, and H. Asada, "Acquisition of human skills for robotic systems," in *Proc. IEEE Intell. Symp. Intell. Control*, 1991, pp. 469–474.

[58] S. Lee and J. Chen, "Skill learning from observations," in *Proc. IEEE Conf. Robot. Autom.*, 1994, vol. 4, pp. 3245–3250.

[59] S. Lee and H. Chen, "Robot skill discovery based on observed data," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, 1996, pp. 2694–2699.

[60] A. Modjtahedzadeh and R. A. Hess, "A model of driver steering control behavior for use in assessing vehicle handling qualities," *Trans. ASME, J. Dyn. Syst. Meas. Control*, vol. 115, no. 3, pp. 456–464, 1993.

[61] E. Fix and H. G. Armstrong, "Neural network based human performance modeling," in *Proc. IEEE Nat. Aerosp. Electron. Conf.*, 1990, vol. 3, pp. 1162–1165.

[62] M. C. Nechyba and Y. Xu, "Learning and transfer of human real-time control strategies," *J. Adv. Comput. Intell.*, vol. 1, no. 2, pp. 137–154, 1997.

[63] M. C. Nechyba and Y. Xu, "Cascade neural networks with node-decoupled extended Kalman filtering," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, 1997, vol. 1, pp. 214–219.

[64] P. Drewes, "Automated student performance monitoring in training simulation," Ph.D. dissertation, Dept. Electr. Comput. Eng., Univ. Central Florida, Orlando, FL, 1997.

[65] M. Georgiopoulos and C. Christodoulou, *Applications of Neural Networks in Electromagnetics*. Norwood, MA: Artech House, 2001.

[66] B. S. Stensrud, "FAMTILE: An algorithm for learning high-level, tactical behavior from observation," Ph.D. dissertation, School Electr. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, 2005.

[67] D. Brunson, *Doyle Brunson's Super System*, 3rd ed. New York: Cardoza, 1979.

[68] D. Sklansky, *The Theory of Poker*, 3rd ed. Henderson, NV: Two Plus Two Publications, 1989.

[69] D. Sklansky and M. Malmouth, *Hold'em Poker for Advanced Players*. Las Vegas, NV: Creel, 2003. 21st Century Edition.

[70] D. Billings, D. Papp, J. Schaeffer, and D. Szafron, "Poker as a testbed for AI research," in *Proc. 12th Can. Conf. Artif. Intell.*, 1998, pp. 228–238.

Brian S. Stensrud (M'06) was born in Orlando, FL, on October 3, 1978. He received the B.S. degrees in computer engineering, electrical engineering, and mathematics (with honors) from the University of Florida, Gainesville, in 2001 and the M.S. and Ph.D. degrees in computer engineering from the University of Central Florida, Orlando, in 2003 and 2005, respectively.

He was a Research Assistant, under Dr. A. Gonzalez, with the University of Central Florida. Since September 2003, he has been with Soar Technology, Inc., Orlando, where he is currently a Research Scientist. His current research interests include knowledge-rich intelligent agents and asymmetric behavior generation.

Avelino J. Gonzalez (F'07) received the B.S. and M.S. degrees in electrical engineering from the University of Miami, Coral Gables, FL, in 1973 and 1974, respectively, and the Ph.D. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, in 1979.

He is currently a Professor with the Intelligent Systems Laboratory, School of Electrical and Computer Engineering, University of Central Florida, Orlando, specializing in artificial intelligence and simulation.

Discovery of High-Level Behavior From Observation of Human Performance in a Strategic Game

Brian S. Stensrud, *Member, IEEE*, and Avelino J. Gonzalez, *Fellow, IEEE*

***Abstract*—This paper explores the issues faced in creating a system that can learn tactical human behavior merely by observing a human perform the behavior in a simulation. More specifically, this paper describes a technique based on fuzzy ARTMAP (FAM) neural networks to discover the criteria that cause a transition between contexts during a strategic game simulation. The approach depends on existing context *templates* that can identify the high-level action of the human, given a description of the situation along with his action. The learning task then becomes the identification and representation of the context sequence executed by the human. In this paper, we present the FAM/Template-based Interpretation Learning Engine (FAMTILE). This system seeks to achieve this learning task by constructing rules that govern the context transitions made by the human. To evaluate FAMTILE, six test scenarios were developed to achieve three distinct evaluation goals: 1) to assess the learning capabilities of FAM; 2) to evaluate the ability of FAMTILE to correctly predict human and context selections, given an observation; and 3) more fundamentally, to create a model of the human's behavior that can perform the high-level task at a comparable level of proficiency.**

***Index Terms*—Context-Based Reasoning (CxBR), fuzzy ARTMAP (FAM), learning from observation, neural network, poker, template-based interpretation (TBI).**

I. Introduction

LEARNING from observation of human behavior is a skill well mastered by human beings, even as young children. Although not all tasks can be fully learned by merely observing others perform (e.g., riding a bicycle and hitting a baseball), many tasks are, in fact, able to be learned by humans through observation (e.g., driving an automobile). In fact, it can be argued that learning from observation shares some commonalities with experiential learning, in that the observer learns from the experience of others. This provides an interesting opportunity for the training of agents to perform humanlike tasks.

There is and has been significant activity in the area of learning from observation in the last several years. We cover that in Section II. This paper describes an investigation into learning the criteria for *context transitions* by observing a player in a computerized game of strategy. To better understand what we mean by a *context* and a context transition, we first present a brief description of *Context-Based Reasoning* (CxBR), which is an essential component of our approach.

Manuscript received May 15, 2007; revised December 6, 2007. This paper was recommended by Associate Editor P. De Wilde.

B. S. Stensrud is with Soar Technology, Inc., Orlando, FL 32817 USA (e-mail: stensrud@soartech.com).

A. J. Gonzalez is with the Intelligent Systems Laboratory, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816-2450 USA (e-mal: gonzalez@ucf.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2008.922062

A. CxBR and Tactical Missions

Webster's dictionary defines context as "…the whole situation, background or environment relevant to some happening or personality" [1]. CxBR, in turn, defines context as previously mentioned, plus the knowledge and functionality for a context-based agent to be able to appropriately act when in this context. In other words, it contains what the agent needs in order to know what to do when in this context. If an agent can identify the context in which it finds itself, it needs only to use the knowledge and functionality defined for that context in order to properly "navigate" it (see [2] for a more detailed discussion on CxBR).

CxBR contexts, in some ways, resemble hierarchical finite-state machines. Indeed, CxBR contexts can be effectively represented by such structures, with contexts roughly equating to states. However, the essential distinction is that contexts in CxBR encompass a grouping of knowledge that is natural (for humans) to a given situation—in effect, anything and everything the agent might need to know while in that context. This knowledge includes functional knowledge (e.g., how to do something) as well as transitional knowledge, which allows the agent to select the most applicable context in a constantly changing environment.

CxBR is an organizational concept and not a language. Contextual knowledge can take the form of one or more functions, rules, neural networks, and simulations, or some form of declarative knowledge. This can drastically prune the search space when looking for relevant operators to address a problem. Grouping knowledge in this fashion can also help in identifying the CxBR context in which the agent finds itself as the transition criteria are defined within each CxBR context (hereinafter called contexts). Traditionally, the contexts have been authored by a knowledge engineer (KE). However, recent research has sought to semiautomatically or automatically build these contexts with the help of knowledge acquisition tools [3] or via machine learning [4], [5]. The work described in this paper is a further effort in the latter approach.

Nevertheless, as the situation experienced by the agent evolves through the natural course of the agent's activity (a game, a mission, a task, etc.), a new set of knowledge may need to be brought to bear ("*activated*") to successfully define and control the behavior of the agent in this new situation. Therefore, recognizing what causes a situation in the environment to change and react to that change by activating the newly appropriate context is not only important but also essential if a system is to correctly perform a behavior. We refer to the criteria that trigger context transitions as the *context transition criteria*. Learning these transition criteria through observation of human performance is the specific objective of the work described in this paper.

We limit our work to problems that involve tactical behaviors. This includes military missions but could also involve team or individual games and other nonconflictive situations where tactical behavior is employed (e.g., driving a car to the airport). The term *tactical behavior*, which is often reserved for behaviors involving military operations, is defined here to denote behaviors with four characteristics.

1) Having a well-defined goal or *mission*.
2) Being characterized by planning and/or maneuvering.
3) Not being well defined as to their execution sequence. Thus, their characteristics may vary greatly across individuals.
4) Needing to intelligently react to unforeseen events or to the actions of others.

B. *High-Level Behaviors*

The overall behaviors learned by our system are considered to be *high-level* behaviors. The precise definition of a high-level behavior is usually omitted in the relevant literature in spite of the fact that their implementation is a primary focus of the work described therein. Jones *et al.* [6] and Jones and Laird [7] refer to high-level behavior when describing the TacAir-Soar system but never explicitly define the term. Likewise, the work reported by Patterson *et al.* [8] describes a method for learning high-level behavior by examining low-level sensors but also stops short of providing a definition of high-level behavior. A common thread found in all of the literature, however, is that the presence of subbehaviors composes the high-level behavior described. In the paper by Jones *et al.* [6], the behavior of piloting a fixed-wing aircraft is described in terms of the composition of its lower level functionality, such as communication and maneuvering the plane.

In the context of this research, we define high-level behaviors as behaviors that can be represented by a sequence of simpler identifiable subbehaviors known as *low-level* behaviors. A low-level behavior is considered to be *atomic* if it cannot be decomposed any further. Otherwise, between high-level behaviors and atomic behaviors at each extreme, there can be several layers of varying levels of behaviors. For example, in the domain of automobile driving, a high-level behavior could be "driving an automobile." Conversely, "pressing down on the accelerator" is considered an atomic behavior. In between, there are such behaviors as "managing traffic lights," "driving in urban areas" (which could indeed include managing traffic lights), "passing," and "turning left."

If it is assumed that each low-level behavior (atomic or not) can be modeled and identified *a priori*, learning is then the process of identifying and remembering the cues (environmental or otherwise) that trigger the transitions between low-level behaviors. The sequence of these low-level behaviors then composes the high-level behaviors executed by the observed human.

We are, furthermore, interested in a class of low-level behaviors that 1) can be identified during observation; 2) exist *a priori* and need not be learned (only recognized); 3) no two such behaviors can be executed at the same time; and 4) are known to be characteristic of the higher level behavior that we do wish to learn to compose.

Behavior B_i, therefore, is learned by determining how our observed human decides to make use of subbehaviors $b_0, b_1, \ldots, b_k$ that compose B_i. Thus, behavior B_i is considered the high-level behavior. The predefined contexts that compose that behavior therefore reflect the low-level behaviors $b_0, b_1, \ldots, b_k$ that together compose B_i.

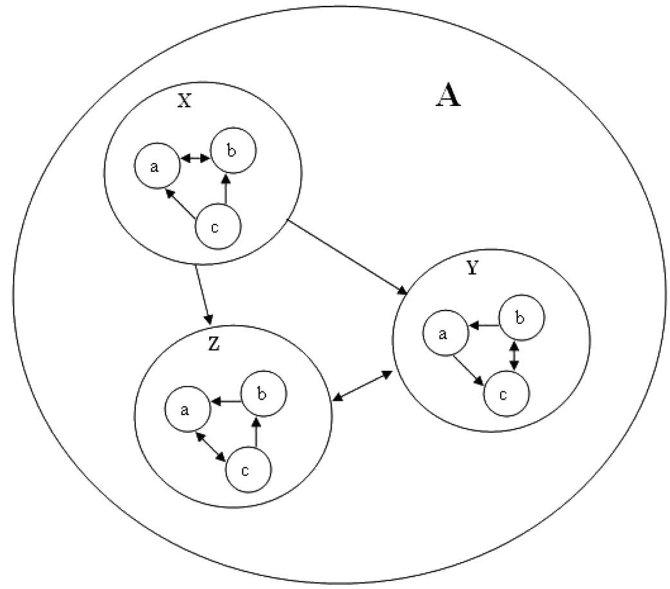

Fig. 1. Learning behaviors by mapping relationships between known subbehaviors.

C. *Example of High-Level Behaviors*

For clarification on our definition of high-level and low-level behaviors, consider the example where behaviors X, Y, and Z are each composed of a set of known lower level behaviors a, b, and c. The different sequences in which a, b, and c are executed in each high-level behavior serves to distinguish them from each other. Our system learns how a human executes behaviors X, Y, and Z (individually) by creating a mapping between the observations of the human's actions and the sequence of the subbehaviors (a, b, and c) that comprise each behavior X, Y, and Z. Assuming that this task is successfully done, an even higher level behavior A can thereafter be learned in the same manner, provided that its execution is composed of a sequence of behaviors X, Y, and Z. A diagram illustrating this point is provided in Fig. 1.

Behaviors a, b, and c are considered to be low-level (in this case atomic) behaviors with respect to behaviors X, Y, and Z. In turn, X, Y, and Z are considered as (nonatomic) low-level behaviors with respect to A.

These types of situations are easily found when we consider tactical human behavior. The task of flying an airplane, as another example, can be broken down into, in the most extreme case, trivial atomic actions—pushing buttons, guiding a control stick in a certain direction, pushing or pulling on the throttle knob, etc. However, flying an airplane is certainly NOT a trivial task. The real knowledge is contained in the processes involved in deciding when to push a particular button, when to pull back on the stick, etc., and in what sequence, depending on the situation at hand. The knowledge is so complex, in fact, that there are hierarchies of subbehaviors that play a role in representing the behavior of flying a plane. Learning to fly is not achieved by learning "buttonology" or stick-maneuvering techniques per se.

It is achieved by learning to execute procedures (e.g., landing, taking off, and maintaining a heading) that involve knowing when to push what button and when and how to maneuver the control stick and/or the throttle.

The argument posed by this example is that, if given the low-level (atomic or not) functionality used by the human, learning his behavior becomes an exercise in identifying a mapping between environmental and situational cues, which we will call *expert stimuli*, and the low-level function or behavior that the human chooses in response to that cue.

D. Observations of Human Performance

In this paper, we describe a learning system that gathers a sequence of observations made of a human performing a high-level behavior. By examining the observations, our system aims to correctly identify the low-level behaviors being executed without feedback from the human and map them to the stimuli within the observations that prompted their selection. With the help of the CxBR modeling paradigm, this system can then be used to develop intelligent models of the learned high-level behavior.

Using CxBR, low-level behaviors are represented as individual contexts, whereas the highest level behavior to be learned is considered to be a CxBR mission. Contexts may contain one and only one behavior (atomic or otherwise) or be composed of several behaviors (atomic, nonatomic, or a combination thereof); which of these is true depends on the context. Some contexts permit only one action to be performed by one atomic behavior. Other situations, however, call for a context that includes more than one behavior although not concurrently.

We define a single *observation* to be a point acquisition of time-dependent inputs used to infer assertions about an agent's environment. We can use time to differentiate and make relationships between two otherwise independent observations. In the following equation, we define an observation $O(t)$ that occurred at time t:

$$O(t) = \langle i_1, i_2, i_3, \ldots, i_n \rangle.$$

Vector $O(t)$ contains fields that represent each input that was introduced to the observer at time t. An observation sequence, therefore, can be considered to be the set of all observations occurring within an arbitrary period of time. The assumption made here is that observations within a time interval occur in discrete points in time rather than continuously. Thus

$$O\{t_0 - t_n\} = \{O\{t_0\}, O\{t_1\}, \ldots, O\{t_n\}\}.$$

As it pertains to our investigation, a single observation includes information about the current environment as well as the current actions of the human. This is critical, because we are attempting to draw a cause–effect relationship between occurrences in the environment and the actions of the observed human. For this research, the learning system develops tactical knowledge from an observation sequence by creating a mapping between an observation pattern and the observed human response. However, it is necessary to process these observations and, from them, learn the knowledge that produces these relationships between the environment and the reaction(s) of the observed human. If we consider these observations as a set of training examples, learning then can be used to generate a knowledge base about actions within the given scenario. Khardon [9] infers a similar definition in his discussion on supervised learning. In our case, however, the learning is to be unsupervised at the input. The observed human does not at all interact with the agent, and learning is done by merely inferring how the human has reacted to his observations. Nevertheless, we define learning from observation as follows:

> *The use of data acquired, through observation, to assert knowledge from which a human's behavior can be intimated.*

We can use our earlier definition of observation to formalize this definition. To do this, we consider the learning process for human E as some function λ of a given observation sequence O_E, i.e.,

$$\lambda\{O_E\} = A_E | A_E = \{A_1, A_2, \ldots, A_w\}.$$

In the preceding equation, the learning algorithm designated by λ operates on an arbitrary observation sequence O_E and outputs a set of assertions A_E that, in some fashion, describe the behavior that has been observed. As the abstraction of "learning" does not imply a restriction in the format of what is learned, these assertions are likewise free to take on various forms: equalities, thresholds, rules, etc.

The potential utility of such a system is twofold. On one hand, the time required to develop acceptable representations of tactical behavior for such agents could be significantly reduced. Instead of producing a complete high-level behavior model by hand, this system could automatically generate what is arguably the most difficult portion of the knowledge: the context transitions.

The second benefit includes the correctness of the knowledge learned. Eliminating a middle person in the development process would conceivably eliminate a source of errors. Furthermore, humans who perform their task with a high degree of proficiency often cannot articulate their knowledge to a third party [10]. A model constructed using a human's introspective explanation can therefore suffer from incompleteness (or even incorrectness) based on this shortcoming. In allowing a system to automatically learn this behavior by observing a human in action, the intermediate step of asking the human to articulate his knowledge is eliminated.

There are, however, some potential caveats in our approach. One is that all contexts and corresponding templates used must be authored *a priori*. This is one significant disadvantage faced by a future developer of an application using this approach. While this is part of the larger problem of knowledge acquisition and machine learning, it nevertheless is quite pertinent to our approach. This paper can indeed serve to reduce the human effort by automatically learning the context transitions. However, significant manual labor is still necessary to prepare the table, so to speak, in order to learn these (e.g., prepare the simulation, run the human subjects, and collect all the observed data). Furthermore, behaviors not predefined as templates cannot be recognized and therefore cannot be learned. These issues are further discussed in succeeding sections.

Before describing our work in greater detail, let us first review the state of the art to see how our work relates to that of others in the field. Given that our application is to poker, we review some of the classic literature on board games and computers.

II. Related Work

Much research can be found in the literature describing learning from observation. While some works address learning high-level behaviors, most focus on learning low-level behaviors. This section describes prior research related to our work.

Board games and computers have a long history together, dating back from the works of Shannon [11], Turing [12], and Newell *et al.* [13]. Charness [14]–[16] studied bridge and chess to identify expertise and their relation to cognitive science. He and his colleagues more recently have used this platform to examine the effects of aging [17]. Certainly, a landmark in computer intelligence was achieved when Deep Blue beat chess Grand Master Garry Kasparov in a chess match in 1997 [18]. This was preceded by important chess playing computers such as HITECH, MEPHISTO [19], and Deep Thought [20], which, prior to Deep Blue, were generally considered to be the best of the chess programs.

Two early researchers of GO playing programs were Zobrist [21] and Ryder [22]. While their work met with partial success, the results of their work could not play as well as a human novice. Additional early work on GO was reported by Kierulf and Nievergelt [23], Kierulf [24], and Wilcox [25].

More to the point, machine learning and board games also have a greatly intertwined history, dating back from Samuel's seminal paper on learning to play checkers [26] and Waterman's subsequent paper on learning heuristics in draw poker [27]. These two seminal works pioneered the machine learning field. Michalski *et al.* appear to be the first to mention observational learning in [28]. Here, they associate learning from observation with unsupervised learning.

In the neural network community, "learning through observation" means that the training data are observations. Fernlund *et al.* [5] define learning from observation as "the adoption of behavior ... through the use of data collected by means of observation." A more descriptive definition describes learning from observation as "inferring concepts by observation" [29]. Here, observation is defined as the act of collecting "characteristics of the relevant environment" [29]. What an observer infers from these observations, however, is a far more complex matter, and so there must be a clear distinction between what is observed and what is inferred about a given environment. One cannot assume that what is reported by a human as "observed" constitutes knowledge that has not already been asserted based on his *a priori* knowledge about his task or scenario. The goal for our learning agent is to develop inferences about "what it sees" based on how a human *reacts* to his observations—not how the human *reports* them. Therefore, observation must be considered as it pertains to the agent—We want to record what the agent sees through the human's eyes. The observations must not, however, include expressions of what the human may annotate or report about his environment.

Sammut *et al.* [30] and Camacho [31] developed systems to observe a pilot's behavior on a flight simulator and implemented the knowledge learned from observation in decision trees. A set of rules was developed as part of the learning process. As part of his work, Sammut coined the phrase "behavioral cloning" to reflect this approach. Sammut's work involves learning rules to perform motor skills involved in flying an airplane. The resulting system learned to fly an airplane as if it were on autopilot in a very strictly defined flight plan. It did not leave room for generalization. Isaac and Sammut's subsequent work [32] extended the previous work to incorporate significant generalization, albeit in a still rather confined domain (maneuvering an aircraft through turbulence).

Sinai and Gonzalez [4] introduced a framework for learning implicit human knowledge through observation of automobile driving behavior within a simulation. Their work is quite relevant to this research because of their attention to partitioning the knowledge by situation (although not called contexts therein). Our work presents almost the opposite approach, in that we assume that the low-level behaviors such as those learned by Sidani and Gonzalez' system (denoted as primitive' in their paper) have already been defined *a priori*. This leaves the actual *situation identification* knowledge to be learned through our neural network approach.

Henninger [33] describes a neural-network-based system that learns how to accurately predict the movement of vehicles in a distributed simulation (ModSAF). Her model builds a predictive model for tank actions by observing a nonhuman but independent algorithm manipulate the tank agent in ModSAF. Gerber [34] employs a *template-based interpretation* (TBI) engine that predicts tank-position information by first selecting its inferred behavioral context. TBI is a method of inferring tactical intent and is likewise essential to our work. It is described in Section III-A. While confined to tank-driving behaviors, Gerber's work is highly relevant to our research. He decomposes the behavior into a set of contexts, which are represented using TBI templates, and using a learning algorithm, he attempts to optimize the identifying weights associated with the templates. The data used in learning is collected from observation of a human-controlled tank. By contrast, the work described in this paper assumes an accurate definition of a set of context templates and attempts to learn the cues that result in a specific context selection.

Johnson *et al.* [35] describe a fuzzy ARTMAP (FAM)-based system that allows computer-generated forces to gradually learn behavior online during a real-time simulation. FAM is reported to have several benefits, including relatively few parameters and the ability to extract and easily explain the results of the learning [36]. FAMs are also essential to our approach.

van Lent and Laird [37] outline the development of KnoMic, a system that extracts knowledge from an expert through observation and then generalizes this knowledge in the form of rules that can be used by an agent to perform a similar task to that of the expert. Whereas Henninger's and Sammut's earlier work focused on learning atomic behaviors from observation, KnoMic is assigned to learn how to execute specific and detailed tasks, such as flying an airplane to a certain destination and in a certain fashion. The authors refer to these types of tasks as performance tasks. As follow-up research to van Lent's KnoMic system, Konik and Laird's work [38] involves the learning of goal hierarchies using inductive logic programming. In the observation mode of this algorithm, the human is again asked to execute a task while annotating goals that he/she has completed during the task. The learning algorithm is then responsible for learning the selection and termination conditions of each goal (when the behavior to execute each goal should be turned on/off). Their use of the human actor beyond demonstrating his skills on a simulator makes their work fundamentally different from ours.

Fernlund *et al.* [5] succeeded in building a system that learned both the low- and high-level behaviors involved in driving a car by observing a human drive a car simulator through a virtual city. Their work used genetic programming to learn individual contexts. Their system generalized quite well and required no intervention by the human actor in the process, beyond performing the behaviors.

Schaal [39] makes a slight distinction between "learning from observation" and "imitation learning." In most cases, learning systems for robots in manufacturing applications try to imitate the exact movement of the human, rather than learn a general behavior. This is typically because, in such applications, the objective of the robot is to imitate the human as closely as possible in a controlled environment.

Walczak and Fishwick [40] describe a study to characterize human expertise by observing the move patterns of chess players. Based on the chunking theory of learning [41], they examine the records of games played by prominent chess masters and a developing player, and compare the chunks learned by these individuals. Their primary objective is not to learn to play the game but to quantify and describe expertise in chess.

Other related work reported in the literature includes that of Pomerlau *et al.* [42], Bentivegna and Atkeson [43], Moukas and Hayes [44], Yang and Asada [45], Floreano and Mondada [46], Pentland and Liu [47], Fogel *et al.* [48], Morrison [49], Crowe [50], Friedrich *et al.* [51], Kaiser and Dillman [52], Rajput *et al.* [53], Hieb *et al.* [54], Gingrich *et al.* [55], Hovland *et al.* [56], Kosuge *et al.* [57], Lee and Chen [58], [59], Khardon [9], Modjtahedzadeh and Hess [60], Fix and Armstrong [61], and Nechyba and Xu [62], [63]. Space limitations prohibit further discussion of these contributions.

Our work differs from the aforementioned works in two ways.

1) We specifically learn the context transitions that are used to link together low-level behaviors into one high-level behavior.
2) We do not interrupt or otherwise consult with the human actor, before, during, or after the learning session. This has the advantage of being able to conceivably learn the behaviors of human actors who do not wish to cooperate with the process (e.g., an opposing team and military enemies). We discuss this in more detail in Section VI.

The works closest to ours is that of Konik and Laird [38] and van Lent and Laird [37] in that they both learn high-level behaviors. However, consultation with the human actor appears to be essential in their approach. Our work represents a different approach to the work of Fernlund *et al.* [5]. Whereas they learn the low-level contexts as well as the transition rules, our work concentrates on learning the transition rules using a vastly different approach.

III. Our Approach to Learning From Observation

Here, we describe an algorithm that identifies low-level (possibly atomic) behaviors when executed by the human and creates a mapping between them and the observations that precede them. The name of this algorithm is *FAM/Template-based Interpretation Learning Engine* (FAMTILE). However, brief descriptions of TBI and FAM neural networks are provided for the interested reader. Readers familiar with these techniques can skip to Section III-C.

A. Template-Based Interpretation

TBI was conceived by Drewes [64] and later enhanced by Gerber [34]. TBI infers tactical intent from observed atomic actions and allows for an inference to be made about the low-level sequence of actions executed by the human and observed by our system. In TBI, contexts are represented by *context templates* or *templates*, which list the expectations of what a human would have to do (in terms of atomic actions) when in the process of carrying out the intended actions. By progressively checking off as "done" the actions that are actually observed, a clearer picture of the intentions of the observed actor comes into focus. Within each template is a set of attributes that indicate actions and conditions; each attribute within a template is considered to be relevant to the context represented by that template. TBI operates by associating a specific observation or observation sequence to the attributes of each template to determine which (if any) of the attributes are satisfied. TBI continuously computes a cumulative score for each template over time. This score is proportional to the number of attributes of a template that are satisfied (Drewes called it "checked off" in his dissertation [64]) and their respective weight. As time passes and more observations are logged and compared to the template's attributes, the cumulative scores of those templates that, in fact, reflect what is happening will tend to rise, whereas those that are irrelevant will either remain low or possibly decrease. At a certain point in time, the template earning the highest score is flagged by the TBI engine as having sufficient confidence that that context is indeed what the observed performer is doing. This process resembles the game of Bingo in many ways. A card is analogous to a template, and a number call to an observation. When a threshold is reached in a specific card (a horizontal, vertical, or diagonal line is checked), success can be declared by yelling "Bingo."

As an example, consider the tactical behavior of driving a car. As a high-level behavior, driving includes several lower level behaviors executed in support of the high-level task: stopping at a red light, passing slower traffic, avoiding and being aware of pedestrians, etc. Oftentimes, there are attributes and cues from the driver and/or from the surrounding environment that can indicate to an observer which atomic behavior is being executed by the driver. For instance, a passenger does not need to ask the driver to indicate when he's attempting to pass a slower car, he can simply look out the window—the driver has changed lanes and increased his speed, the passed car is driving too slow, etc.

In TBI, we consider these cues to be the attributes of a context and group them together within a context template. These attributes are then assigned a weight indicating their importance in identifying the context. Because the behavior expected within each context is known *a priori*, creating templates with useful attributes is a reasonable task for a KE.

B. FAM Neural Networks

FAM is a neural-network clustering technique developed at Boston University in the early 1990s. The network was introduced by Carpenter *et al.* [36] and is described in detail by

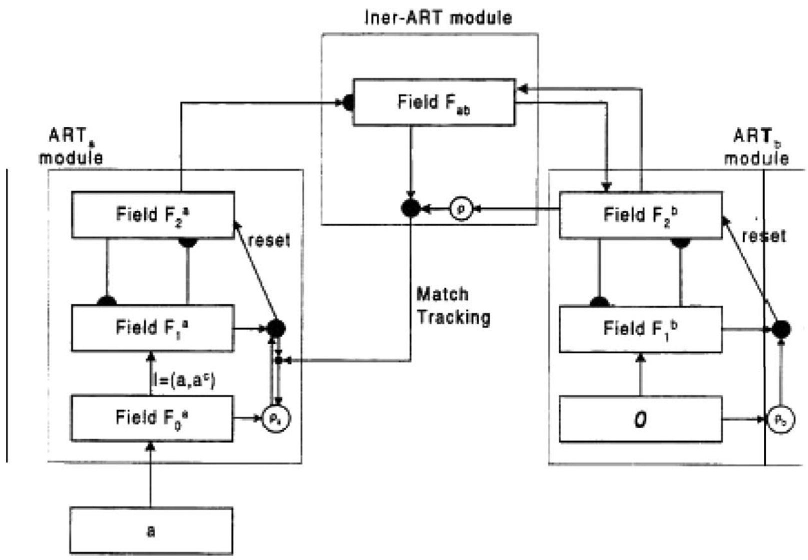

Fig. 2. Block diagram of a FAM architecture [65].

Georgiopoulos and Christodoulou [65]. The goal behind this technique is to produce a neural network that is proficient at dealing with "misbehaved" batches of test patterns, i.e., patterns where a minority of the testing patterns share little in common with the majority used to train the neural network but are equally (if not more so) relevant.

A block diagram of the FAM architecture is provided in Fig. 2. The ART$_a$ and ART$_b$ modules within FAM are responsible for generating pattern templates that correspond to a certain pattern form, essentially dividing the pattern set into clusters. Each template created within the ART$_a$ module represents an input-pattern type that corresponds to a specific output template created by the ART$_b$ module. The Inner-ART module is then responsible for creating a many-to-one mapping between the templates within ART$_a$ and those within ART$_b$.

For example, consider a situation where a neural network is trained to recognize alphabetical letters when seen and, in response, produces a specific sequence of numbers based on the letter input. When training a FAM module, the ART$_a$ module is responsible for learning to recognize each input letter, whereas the ART$_b$ module is responsible for learning to recognize each output sequence. The Inner-ART module creates the mapping between specific letters and their corresponding output sequence.

C. Our Approach

The FAMTILE algorithm is composed of two major parts: Part 1 involves inferring the context being experienced by the human actor being observed. Part 2 relates to mapping the context inferred in part 1 to the environment to determine the potential causes of a context transition. Part 1 employs the aforementioned TBI algorithm, whereas part 2 employs FAM neural networks. These two parts are independently discussed.

After learning the set of conditions that trigger atomic behavior transitions, a CxBR model that reflects the high-level behavior of the human observed during the simulation can then be constructed. This model contains both the low-level contextual knowledge developed *a priori* and the knowledge learned by this system that identifies when each low-level context becomes activated. We begin this section by defining terms and discussing how the observational data are captured.

1) Acquiring the Observational Data: Before the learning process can begin, the human actor to be observed must clearly understand the mission he is to perform. He must also be in an environment (either live or simulated) that he can affect through his actions. Furthermore, the observational system must be situated so it has the most direct access to the stimuli seen by the human actor without impeding him in any way. In this paper, we simplify the problem somewhat by using a simulator to implement the learning algorithm. This facilitates the observation process and allows us to concentrate on the technical feasibility of the algorithm.

While the human actor executes a high-level mission within the simulation, FAMTILE records all relevant and visible stimuli on the human, along with the actions taken by the human at the time those stimuli are presented. A recording is made at each decision point i reached during the execution of the behavior to be learned. In the simulated world, these decision points can be either continuous points or segments of time or planned decision points where time is not relevant, such as in a turn-based game, such as chess or poker. To account for the reactive nature of the human's actions at any decision point i, we refer to the time at which the stimuli are presented as time i^- and the time at which the human switches his active context as time i^+. We assume that the human cannot anticipate the environmental trigger but must perceive it before acting to switch contexts. Anticipation is a complicating feature at this time, and we leave that for future research. However, we see no fundamental impediment to a future implementation of this feature.

At the point when the human completes the scenario, the learning system will have compiled a set of recordings that should encompass all relevant stimuli and the actions taken by the human actor. This set is known as the *observation sequence* for the executed scenario. Individual members of this sequence are distinguished by the simulation-time at which they were recorded and are referred to, naturally enough, as *observations*.

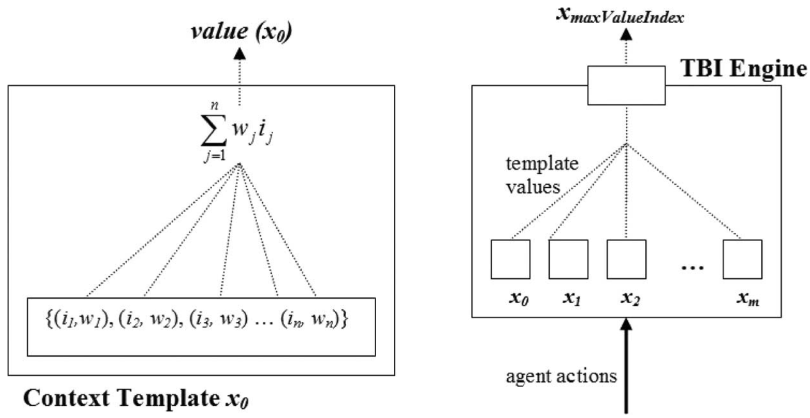

Fig. 3. Generic context template and the TBI engine.

These observations, which are labeled σ_i, denote decision point i, along with the set of visible stimuli Φ that existed at i^- and the set of actions Γ taken by the human at i^+. Thus

$$\sigma_i = \langle \Phi_{i^-}, \Gamma_{i^+} \rangle$$

where $\Phi_{i^-} = \{o_0, o_1, \ldots, o_n\}$ are the traits of observation i, and $\Gamma_i = \{j_o, j_1, \ldots, j_n\}$ represent the actions taken by human in response to observation at i.

We define the complete observation sequence Ω_n to be the set of observations σ_i taken of the human throughout an entire scenario n, i.e.,

$$\Omega = \bigcup_i \sigma_i.$$

After the observations of the human are complete, the entire observation sequence Ω is presented to FAMTILE. At this point, the actions of the human are interpreted by the TBI engine, which will convert Ω_n into a new observation sequence $\Omega_n^{'}$, where the set of actions taken (represented by σ_i in Ω_n) are replaced with the interpreted context. This context, which is inferred by TBI for decision point i, is represented by Ψ_{i^+} in the following equation:

$$\sigma_i^{'} = \langle \Phi_{i^=}, \Psi_{i^-} \Psi_{i^+} \rangle$$
$$\Omega' = \bigcup_i \sigma_i^{'}.$$

In addition, represented within $\sigma_i^{'}$ is the inferred active context of the human prior to decision point i. This context is denoted as Ψ_{i^-} and is identical to the context inferred from the previous decision step Ψ_{i-1^+}. FAMTILE's TBI engine achieves this transformation by making an interpretation of each atomic action. Prior to the observation time, a KE defines each atomic behavior (i.e., the behavior the system will observe) that is necessary for the execution of some high-level behavior (the behavior the system will infer). From these specifications, the KE also creates a set of context templates. Each of the templates' attributes is derived from fields within observation σ_i.

Now we move on to the first part of the FAMTILE process: how to infer the human's context.

2) Part 1—Inferring the Context of the Human Performer: We assume that all low-level behaviors can be identified through observation. Because the low-level behaviors that compose a particular context are known, we need only recognize them through observation and record their presence. Then, we must put them together into a sequence that explains the higher level intentions (i.e., the context) of the observed performer. To accomplish the latter case, we employ the TBI technique discussed in Section III-A.

For convenience, we will consider an arbitrary set of contexts $C = C_1, C_2, \ldots, C_n$ and corresponding set of templates $T = T_1, T_2, \ldots, T_n$. Using this representation, we say that a template T_j includes all attributes and weights common to its corresponding context C_j. In a given scenario, all contexts C_i are represented within TBI by a specific template T_i that defines the attributes of C_i.

Each attribute a_i in template T_j is a representation of a condition that is prevalent in context C_j. Weight w_i represents the importance of a_i in determining context C_j. A low weight value for w_k indicates that attribute a_k is not an essential or even very important characteristic of context C_j. Conversely, a high value for w_m indicates that attribute a_m is highly relevant, perhaps even essential, for context C_j. This representation was used in both the works of Drewes [64] and Gerber [34]. Thus

$$T_j = \{\langle a_0, w_0\rangle, \langle a_1, w_1\rangle, \ldots, \langle a_n, w_n\rangle\}.$$

The TBI engine infers a context by first evaluating the *state* of each attribute in its set of predefined templates. After each attribute is assigned a value (typically T or F, depending on whether that action has been observed or not), a weighted sum is computed for each template T_j and used as its *template score*. This template score s_j is computed as follows:

$$s_j = \sum_{i=0}^{n} a_{ij} w_{ij}.$$

The value assigned to each attribute a_i in template T_j depends on the nature of the attribute. Fig. 3 represents a TBI engine that considers a set of m context templates and n attributes per template. On the left side of the figure, we see the composition of a generic context template score. Note that the score is generated using a simple weighted sum of each attribute score (computed using the preceding equations). The right side of the figure illustrates the comparative portion of the engine—each score is reviewed and the maximum score is selected. The context associated with $s_{\max}$ is chosen as the inferred context for that observation. Stensrud [66] provides a more thorough description of how TBI is applied to FAMTILE. The output of this first part, therefore, is an indication of what context the human is experiencing while the system observes his actions.

3) Part 2—Associating Context Change to Environmental Triggers: This section discusses the part of the FAMTILE algorithm that learns the transitions between contexts affected by the human performer. It accomplishes this through neural networks.

The ability of a neural network to handle "misbehaved" training sets is of particular relevance to learning from observation. Consider the knowledge required to drive an automobile, which is an example of a tactical skill. The ability to handle a tire blowout while driving, particularly when at high speeds, is certainly important. However, this skill is rarely required, simply because tires rarely ever blow out. If one were to observe an automobile driver in order to train a neural network how to drive, the training pattern corresponding to a blown-out tire would represent a very small minority of the training set.

In a CxBR model for tactical control of an entity in a simulation, it is possible that important events requiring a specific context transition infrequently occur. Because of this, training patterns representing these types of context transition cues will most likely be underrepresented within a training set. In such situations, traditional neural networks have a difficult time learning these patterns as a result of the strong emphasis on the other patterns. In these cases, the neural network tends to "overlearn" the more frequent patterns and discard the others as noise within the training set. In the case of our work, this noise may represent an interesting and important observation, making the human's response to it very important to record. FAM neural networks are adept at recognizing the infrequent patterns without reversing the knowledge of any well-learned patterns [65].

Through the creation of clusters, FAM also has the ability to handle a large sample of training patterns necessary for a complete observation of a human's behavior. This clustering process has the effect of significantly reducing the complexity of a decision space, based on the size of the clusters created. The advantage here can be visualized by again considering the task of learning driver behavior. Because recording a decision-making cue (e.g., to change lanes, to brake, and to turn) often requires fine granularity across observations, several hundred observations of the driver/expert may be recorded throughout a few-minute driving task. Furthermore, values for the driver's speed, heading, distance to other vehicles, and other potentially significant factors will certainly fluctuate, at least nominally, along a several-second interval where no significant behavioral change is executed. This is not because the driver consciously decides to make these changes (decisions that should be recorded and learned) but simply because of the dynamics of the environment and the driver's inherent inability to hold an identical speed and course. A FAM system allows for nearly identical input patterns such as these (that map to the same output) to be represented by a single cluster. By creating a less complex decision space, we significantly reduce the order of the learning task.

Our specific learning objective here is the transitions between contexts. The new context would contain the appropriate functionality to allow the agent to properly manage it. FAMTILE is built to recognize and capture those triggers and learn them for subsequent use by the agent. We assume that all other functionality—that which permit a context to correctly control an agent when active—is already known *a priori*.

Set Ω' is, at this point, transformed into a form usable by FAM. This operation is done by converting each $\sigma_i^{'}$ into a single training pattern. For a training pattern to be readable by the FAM neural network, each field must be a *fuzzy value* (some real number between $[-1, 1]$). Within FAMTILE, the input portion of the training pattern is derived from $\Phi_{i=}$ and Ψ_{i-}, whereas the output pattern is derived from Ψ_{i+}.

The subset $\Phi_{i=}$ of observation sequence $\Omega_n^{'}$ consists of fields representing the human's complete observation at time i^-. The human's active context at i^- is denoted by Ψ_{i-}. Converting the observation for Ψ_{i-}, the observed active context at i^- involves the same procedure, regardless of the scenario. To convert the identified active context into a field within the input pattern, one field is set aside for every possible context in the scenario. If a context j is identified as the active context, the jth field is assigned a value of 1, and the other "context fields" within the input pattern are assigned a value of 0.

This is done to persuade input patterns with different active contexts to bind to different templates in ART_a. The following equation represents an arbitrary input pattern converted from $\Phi_{i=}$ that can be presented to FAM, which we refer to as $\dot{\Phi}_{i=}$:

$$\dot{\Phi}_{i=} = \overbrace{o_1, o_2, o_3, \ldots, o_{k-1}}^{\text{observation fields}}, \underbrace{c_1, c_2, c_3, \ldots, c_{n-1}}_{\text{active context}(n-1)}.$$

Output pattern Ψ_{i+} is simply a representation of the inferred active context at i^+. Because of this, Ψ_{i+} can be represented as a j-bit binary number to identify one of j distinct contexts as active, just as is done for the inferred context at i^-. Within Ψ_{i+}, all bits are set to 0, except for one. If that one set bit is the ith bit (i.e., oc_i in the expression for $\dot{\Psi}_{i+}$), that means that context i has been identified as the active context for i^+. This representation scheme will make for a trivial clustering task for ART_b, because exactly one output cluster will be generated per context. Representing a context name in this manner allows for the output of ART_b to be both readable and unambiguous for either a KE or a separate module created to read its output. The following equation represents an arbitrary input pattern converted from Ψ_{i+} that can be presented to FAM, which we refer to as $\dot{\Psi}_{i+}$:

$$\dot{\Psi}_{i+} = oc_1, oc_2, oc_3, \ldots, oc_{n-1}$$
$$\text{(a bit string representing the selected active context)}.$$

The input and output patterns $\dot{\Phi}_{i=}$ and $\dot{\Psi}_{i+}$ presented to FAM reflect observations recorded at specific times during the scenario, along with the active contexts at those times, as identified by the TBI engine. The input patterns are represented by quantitative values for each stimulus on the human—enemy movements, environmental conditions, current physical conditions, etc. The output patterns represent the action taken by the human in response to the input pattern presented, where each action reflects a transition from the provided context at the input to a new active context which is inferred using TBI. The implication here is that every action (and thus every output pattern) will

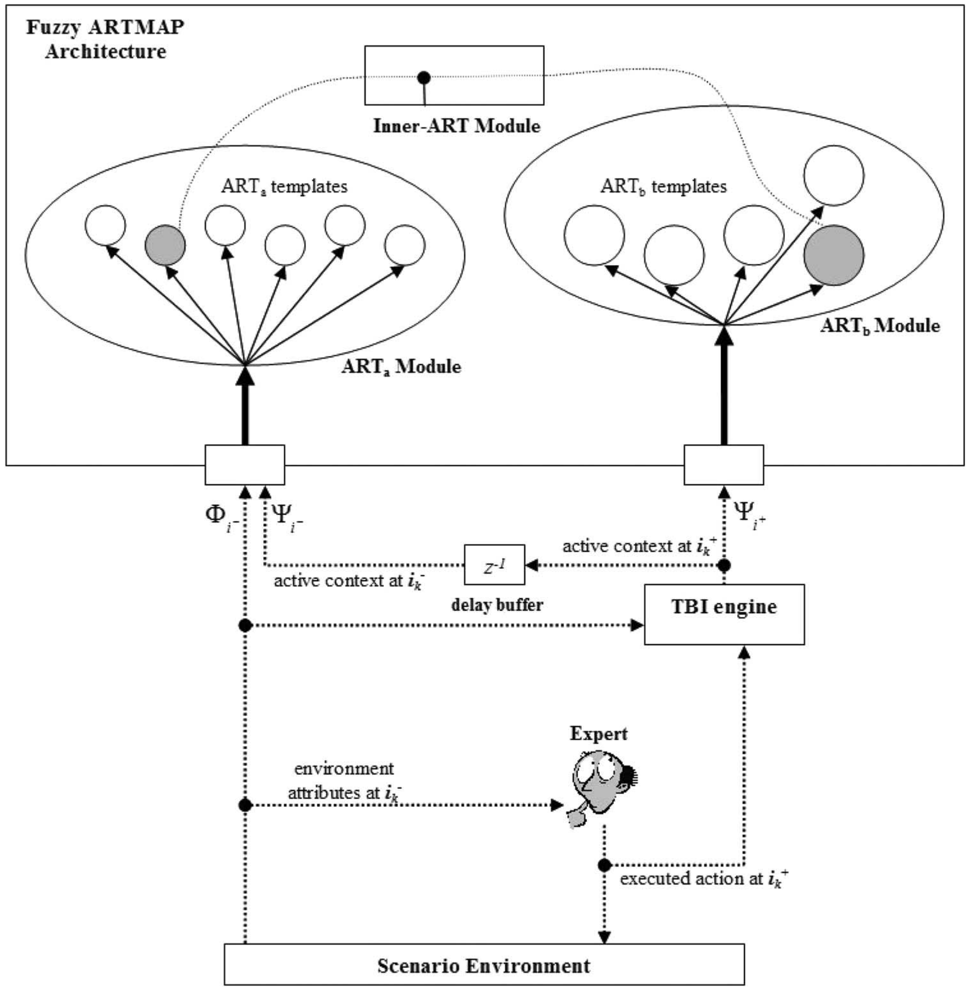

Fig. 4. Learning context transitions in FAMTILE.

represent a transition to a new context, which is of course not always the case. Rather, actions representing no context transition are also represented by patterns that require a transition to the current context—the equivalent of no context change.

A training pattern is generated and presented to FAM for each observation made of the human during the execution of a scenario. Learning occurs through the creation of clusters in the ART$_a$ and ART$_b$ modules and of a many-to-one mapping between those templates. ART$_a$ templates represent clusters of input patterns, similar in their representation, to which the human has responded by making a specific context transition. That transition is stored in a template in the ART$_b$ module, and a mapping between the two templates is created. When the network subsequently encounters an input that matches the input pattern cluster represented by that template in ART$_a$, it will know that the appropriate response is stored in its mapped template in ART$_b$.

Fig. 4 illustrates FAMTILE in learning mode. A recorded observation includes both the stimuli on the human and his resultant decision. A decision is considered to be the action made by the human in response to a set of stimuli presented at i and is expressed as the context that the agent enters (makes active). These stimuli, along with the active context in which the human is operating at i^-, constitute the input pattern that is presented to ART$_a$. The actions that the agent executes in response to these inputs (at i^+) are analyzed by a TBI module, which then outputs the most likely candidate for the context that corresponds to those actions. That context name is then presented to ART$_b$ as the output pattern for i and is also stored for the next decision-point $i + 1$, where it will be presented as part of the input pattern as the active context prior to the stimuli presented and actions taken at $i + 1$.

The task for FAM, then, is to learn the correct context transition, given the current active context and the input stimuli on the agent. To do this, the network will create templates in ART$_a$ that effectively cluster similar input patterns that induce a specific context transition by the human. The template corresponding to the actual transition made will be stored in ART$_b$, and the Inner-ART module will create a link representing a mapping between the two templates. After the training phase is complete, there will exist a many-to-one mapping between the input-pattern templates in ART$_a$ and the context transition templates in ART$_b$.

D. FAMTILE Operation

A summary of the sequence of events required for the FAMTILE algorithm is presented here.

1) The human actor executes a high-level behavior in some simulation or simulator.

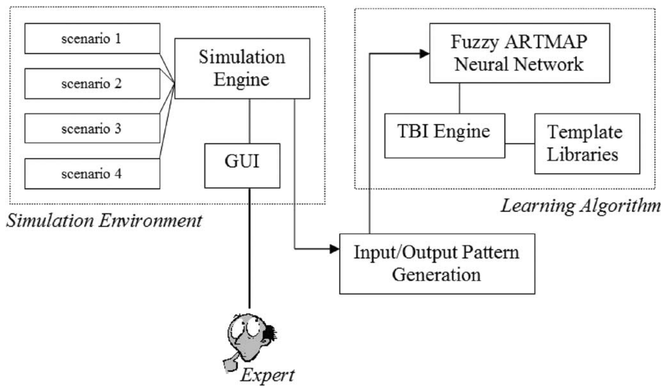

Fig. 5. Block diagram of the testing environment.

2) FAMTILE collects an observation sequence of the human's actions.
3) The TBI engine interprets human actions and infers corresponding contexts.
4) The observation sequence with contexts inserted is converted into a set of input patterns.
5) The sequence of contexts is converted into output patterns.
6) The input/output patterns are paired and presented as training patterns for the neural network.
7) The neural network is trained to recognize observation patterns and map them to specific high-level contexts.

IV. Test Prototype

To evaluate the FAMTILE concept, a prototype system was built. However, in evaluating this prototype, it was first necessary to construct a test bed simulation in which training vignettes could be developed and executed. This simulation was written in Java and was designed to interface the FAMTILE prototype with the testing vignettes and to provide a graphical user interface for the human actor to perform his behaviors. A block diagram of the simulation environment is provided as Fig. 5.

The simulation engine provides both the logic of the vignettes and their graphical user interface, which was developed in Java. This interface was created in an attempt both to attract human test subjects to participate and to provide them with as realistic a vignette as possible.

The simulation engine implements the logic and execution engine for each of the four vignettes. When a human subject selects one of them, the simulation instantiates it and presents the human with his first decision point. Each vignette is such that the human actions are *turn based*, and observations for a certain decision step represent a set of stimuli and resultant action for one turn. In a turn-based simulation, decision steps are triggered on human actions and not on actual clock time. This property ensures for FAMTILE that the human is making decisions in response to a known set of observations and that there is a correct pairing between those observations and that action. Otherwise, the system could not guarantee that the human was making decisions based on the observation recorded for that corresponding time step. The actions that take place within the simulation during training mode are presented here.

- The simulation prompts the human actor to enter his/her name.

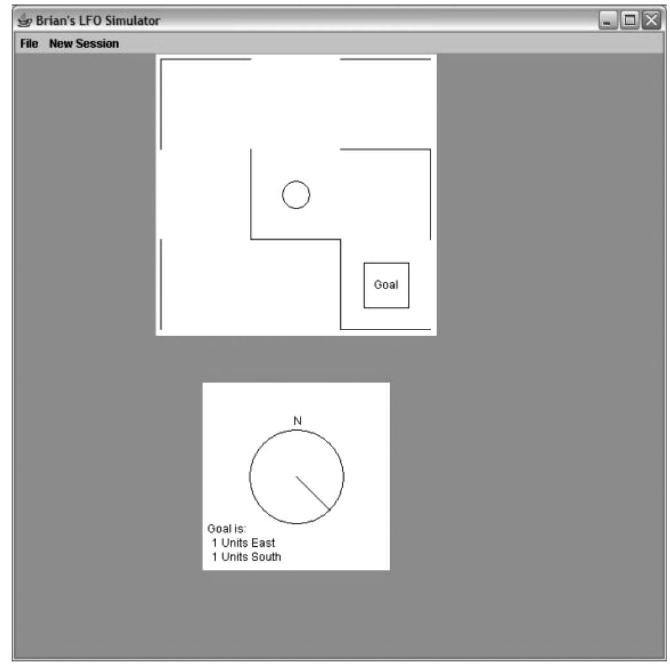

Fig. 6. Vignette A.

- After the name is entered, the human selects a training vignette.
- When a vignette is selected, the simulation engine calls the initial commands that begin that vignette. That vignette then displays the situation for the human and then pauses until the human has made his/her response.
- That response triggers an event in the simulation that brings up the next situation and writes the stimuli/response pair to a text file, which is read by the interface class after the training session.

To make a thorough evaluation of the learning algorithm, four different test vignettes were developed. These are based on two behaviors: 1) moving within a maze environment and 2) playing a game of poker.

A. Maze Navigation: Vignettes A and B

The first two training vignettes involve the navigation of a 2-D maze. For each vignette, the human is asked to navigate from his position within a virtual maze to a specified goal position. At each point during the vignette, the player is provided a compasslike directional icon that indicates the distances—in both the x and y directions—to the goal position. If the goal position is located within the player's field of view, its position is marked on the map.

In Fig. 6, the circular shape occupying the center position in the maze indicates the position of the human's avatar. In vignette A, the player can only see one space in all directions from the avatar's position. From the observations of this figure, the human makes a decision on which direction to move. In this vignette, the avatar and goal positions are reinitialized after each human action.

In vignette B, the human is asked to navigate the avatar toward a goal position and is given a larger frame of view (see Fig. 7). The simulation also records the spaces that have been visited by the avatar along his path to the goal position and marks these spaces with a square shape on the maze view.

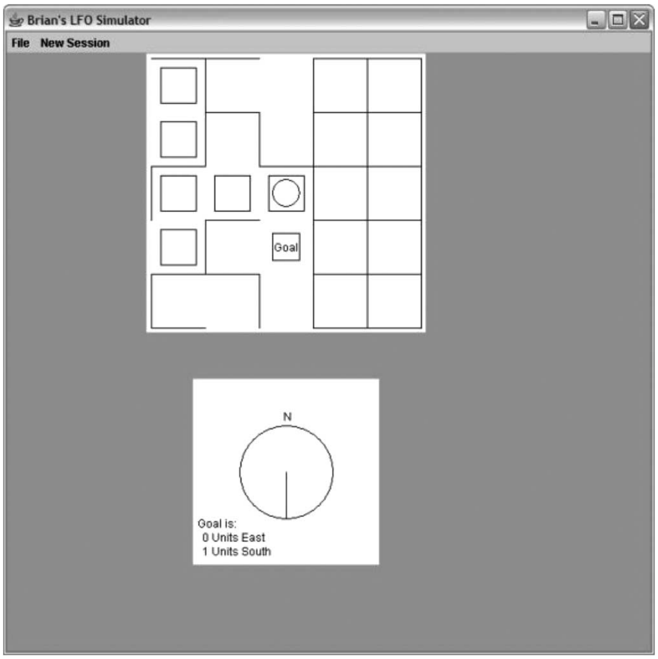

Fig. 7. Vignette B.

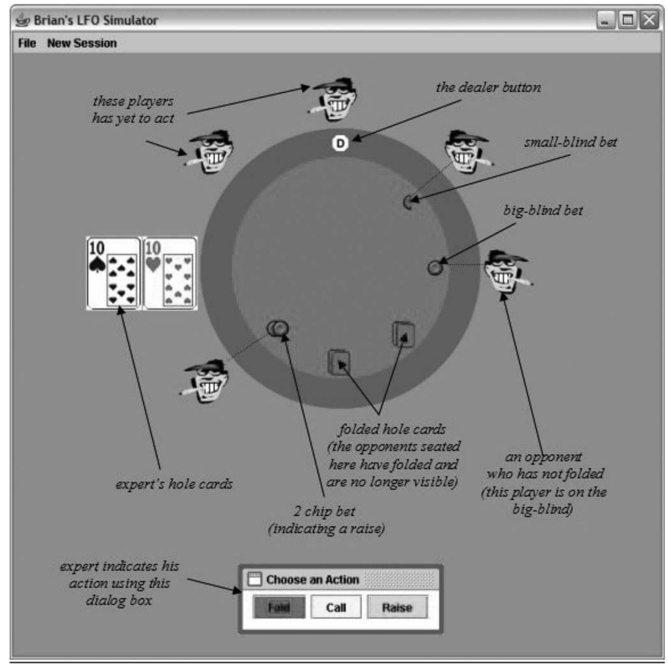

Fig. 8. Vignette C.

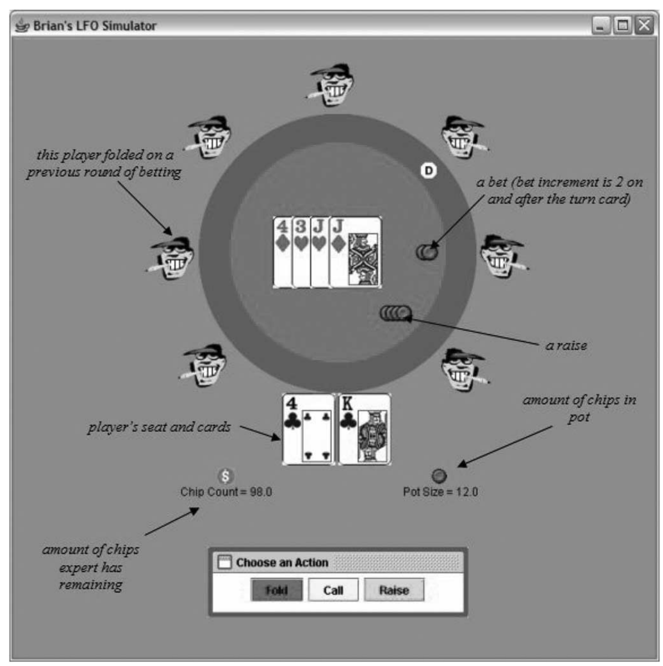

Fig. 9. Vignette D.

For vignettes A and B, no context templates are required, because there are no contexts implied with the human's movement. Vignettes A and B are used to provide control cases to evaluate the ability of the FAMs to learn without the encumbrance of the FAMTILE system. More details on this are provided in Section V.

B. Poker Game: Vignettes C and D

The other two training vignettes involve the game of Texas Hold'em Poker. The succeeding sections assume basic understanding of the concepts of poker and the Hold'em Strategy [67]–[69]. These vignettes are used to evaluate the ability of the entire FAMTILE algorithm, including recognizing the atomic actions of the human.

For this paper, two training vignettes were developed using the Limit Hold'em game. In the first poker vignette (vignette C), only one betting round occurring prior to the *flop* is considered. The human is placed at a random position at a poker table and seated with seven computerized opponents. The dealer button is placed at a random position, and each player is dealt two hole cards. Starting with the player to the left of the big-blind bet, each opponent makes an action (either to fold, call, or raise) until it is the human's turn to act. At this point, the human will know his two hole cards, his position at the table, and the actions of each opponent who has acted before him. The simulation then prompts the human to make an action: either to fold, call, or raise. The human's actions are recorded, along with all applicable observations at that point; then a new hand is dealt, and the player is reseated. This process continues until the simulation has collected a requisite number of observations. A screenshot of the simulation for this vignette is provided in Fig. 8.

For the second poker vignette (vignette D), the human is asked to make decisions throughout entire hands and accumulate chips throughout the vignette. This is depicted in Fig. 9. This vignette begins just as the first poker vignette—the human is placed at the table with seven opponents, and the button is placed at a random position at the table. A hand is dealt, and each opponent makes an action on their cards until it is the human's turn to act. When the human acts, however, the betting round continues as well as the hand and proceeds just like a standard round of Limit Hold'em. After each round, the dealer button rotates one chair to the left, and a new hand is dealt. A *chip count* is stored for the human, which reflects the amount of money won/lost during the sequence of hands played.

In this vignette, the situations encountered by the human are far more robust and are designed to challenge his playing

TABLE I
RAISE IN POSITION CONTEXT

playerAction = Raise	weight = 6
distanceFromButton = 0	weight = 3
numPlayersInPot = 2	weight = 0.5
numBetsToCall = 1	weight = 0.5

ability. Because the vignette involves entire rounds, the opponents at the table react to the human's decisions and use many of the strategies outlined in [69] to try and win hands. Since this vignette involves the observation of humans playing against opponents, it was important to create opponents who are able to pose at least minimal challenge. Opponents for the vignettes are programmed with the following:

- basic understanding of the strength of its hole cards before the flop;
- basic understanding of the hand strength relative to the cards on the board;
- basic understanding of the hand potential relative to the cards on the board;
- ability to bluff;
- ability to trap or slowplay;
- ability to change play based on position and amount of action in the betting round.

For these vignettes, each action taken by the human must first be interpreted by the TBI engine before presenting a corresponding output pattern to the FAM. This output pattern is the context of the action taken, as interpreted by TBI. Individual actions performed by the human are assumed to be a consequence of the human acting in a particular context. To make an interpretation of the context embodied by the human's recorded action, the TBI engine matched each template against the appropriate conditions present in the observation. The engine then infers the context in which the human is likely to be acting. This determination is then recorded by the interface module and transformed into a bit sequence representing the output pattern for FAM using the technique discussed in the previous section.

In vignettes C and D, we consider a context to be a circumstance and/or rationale for making a particular play. The *raise* action, for instance, is divided into contexts that differentiate the inferred reason for the raise. As discussed by Sklansky [68], there is a variety of purposes behind making a raise: to force weaker hands to fold; to get more money into a pot; to bluff, thereby causing stronger hands to fold; etc. While the human's intent cannot be recorded through strict observation, it can be inferred if each of these purposes is encoded by a context. Using expertise gathered from poker experience and from various texts [67]–[69], a set of contexts that result in each possible action (e.g., raise, call, bet, and fold check) in both vignettes was generated. When an observation is presented to FAMTILE's TBI engine, it is compared against the attributes of each context template and generates a score for that template. Consider the template in Table I for the *RaiseInPosition* context. This context refers to a situation where the human has made a raise based mostly on his strong position relative to the dealer button. As stated earlier, players *on the button* get to act last on each postflop betting round, giving them a significant advantage of being able to react to each opponent's play.

Note the weights associated with each attribute. The most heavily weighted attribute is the player's action: if the player does not make a raise, this weight induces the TBI engine to calculate a low score for this template. The other weights are assigned based on their relevance to the context, i.e.,

$$score_{\mathrm{att}} = \frac{(1 - |att_{\mathrm{observed}} - att_{\mathrm{template}}|)}{range_{\mathrm{att}}} weight.$$

Since the training patterns for the neural network come directly from the observations of the human under study, the amount of diversity among those training patterns is completely dependent on the robustness of the vignette in which that human operates.

Knowledge used for training can only be extracted from observations. Thus, any relevant knowledge not executed within an observed simulation will not be learned by the neural network. Because of this, there will be gaps in the tactical knowledge about situations not encountered by the human during the observation phase. If these gaps are ignored by the learning system, the resultant autonomous agent will have no intelligent response if presented with that unlearned situation. The only defense against these gaps in knowledge is to train the network with as many examples as possible in hopes that they sample as much of the human's knowledge as possible, i.e., provide vignettes in which the human must use all or most of his/her tactical knowledge.

C. *Generating Training Inputs from the Observation*

Generating training points for the maze vignettes is a matter of placing the player and goal at random locations within a fixed maze. Each time the player makes a move, the next training point input pattern becomes either a new random position for both him and the goal (as in vignette A) or the updated maze state based on the direction of the player's previous movement (as in vignette B). The output pattern for that training point is then the action taken by the expert for the corresponding maze state represented by the input pattern. Each of these patterns, however, must first be translated into a readable form, so that they can serve as useful training patterns for FAMTILE. The output pattern is simply the context that the expert has chosen as a response to the stimuli represented by the input pattern.

For the Poker vignettes, the simulation must generate and record the following pieces of information for each observation:

- player's hole cards;
- board cards (vignette D);
- player's position;
- position of the button;
- opponent actions;
- amount of money in the pot (vignette D);
- player's action.

To generate this information, the simulation deals a random hand to the expert and seven automated opponents. Each opponent makes an action until it is the player's turn. At this point, the state of the hand is recorded, along with the action made by the player for his turn. For vignette C, each of these points occurs during the betting round prior to the flop.

For vignette D, this observation is expanded to include interpreted information about the player's hand and position relative

TABLE II
VIGNETTE D CONTEXTS

foldWithWeakHand	Player folds because his cards are weak
foldWithMediocreHand	Player folds an average hand
foldWithDrawingHand	Player folds a good drawing hand
foldWithStrongHand	Player unknowingly folds a strong hand
checkWithWeakHand	Player checks with a weak hand, likely with the intention to fold if there is a bet made
checkWithDrawingHand	Player checks a hand that is on the come to a possible winning hand, and would like to see another card for little to no money
checkWithMediocreHand	Player checks with a marginal hand, likely to observe the action at the table
checkWithMonsterHand	Player checks with a monster hand, to fake weakness and induce action from his opponents
checkWithStrongButVulnerableHand	Player checks with a strong hand that is vulnerable to drawing hands
callWithWeakHand	Player makes an extremely loose call with a weak hand
callWithMediocreHandContext	Player makes a 'loose call' with a hand that 'tighter' players would likely fold. A 'tight' player typically only plays with very strong hands and draws.
callWithDrawingHand	Player calls with good multiway hole cards to see a flop, or if he is on a good draw (to a flush, straight, etc.)
callWithMonsterHand	Player calls with a monster hand, attempting to slow-play his hand
callWithStrongButVulnerableHand	Player calls with a strong hand vulnerable to drawing hands
betWithWeakHand	Player bets with a weak hand to bluff
betWithMediocreHand	Player bets with a marginal hand, either to bluff or to induce a weaker hand to fold
betWithDrawingHand	Player bets a drawing hand on a semi-bluff.
betWithStrongButVulnerableHand	Player bets with a strong hand vulnerable to drawing hands
betWithMonsterHand	Player bets with a nearly unbeatable hand
raiseWithWeakHand	Player makes a raise with a weak hand in order to induce the table to fold (a bluff)
raiseWithMediocreHand	Player makes a raise with a mediocre hand, either to bluff or to induce a weaker drawing hand to fold
raiseWithDrawingHand	Player makes a raise with a strong drawing hand, in an attempt to induce either folds or 'free cards' in later rounds.
raiseWithStrongButVulnerableHand	Player makes a raise with a strong hand that could get drawn out on
raiseWithMonsterHand	Player has a nearly unbeatable hand, and is raising to extract the most amount of chips out of his opponents

to the rest of the table. To do this, the following parameters are used:

- *hole cards:* rank of the player's two hole cards (both are scaled to values < 1);
- *suited*: boolean value indicating whether cards have the same suit;
- *hand strength*: fuzzy value of the player's hand, as calculated in [70];
- *pPot*: fuzzy value representing the potential of the player's hand drawing to a winning hand [70];
- *nPot*: fuzzy value representing the potential of the player's hand decreasing in strength due to future board cards [70];
- *betting round*: 4-bit binary value representing the current betting round;
- *last action*: 4-bit binary value representing what the player did on his last turn to act;
- *pot size*: number of chips currently in the pot, scaled to a fuzzy value < 1;
- *opponent bets in pot*: scaled to a fuzzy value < 1 by the size of the largest bet.

Table II summarizes the contexts used for vignette D. There are a total of 24 contexts. For vignette C, only 12 contexts were used. This is because there are fewer actions available to the player in vignette C (player cannot bet) and, more importantly, the player has less information about his hand (no board cards are shown in vignette C, only preflop action) and therefore cannot classify the situation to the same level of granularity.

When the simulation records the expert's action during the observation, the result is simply a character value representing either a raise, fold, or call. For both poker vignettes, however, FAM is used to create a mapping between the observed situation and the expert's choice of context, and not simply his action. To make this transformation, the interface extracts necessary variables from the input pattern to present to the TBI engine, which makes a prediction of the most likely context that the expert has chosen. For vignette C, there are 12 contexts from which the expert can select.

An output pattern for vignette C would therefore be a 12-bit binary number with all but one number set to zero. That number, in the jth position, represents that the TBI engine has identified context j as the active context for the observation represented by the input pattern.

In this prototype, the FAM clusters are stored as 1-D arrays—one for each cluster in the ART_a and ART_b modules. Each entry in these arrays represents a field value of that cluster. To store the mappings, a separate array is created that represents the InnerART module of the FAM. This array contains one field for each cluster created in ART_a. The value stored in each field is the index of its mapped cluster in ART_b. For instance, if the ART_a cluster i is mapped to cluster j in ART_b, the InnerART array would look like $[ia_1, ia_2, \ldots, ia_c = j, \ldots]$. Here, the field containing the value j is stored in the ith slot.

D. Comments on the Application Selected

Two issues that demand some discussion and further explanation come to mind. We address these in this section.

The data obtained were observed from a simulation of games, rather than from watching humans play the game in the real world. This is particularly true for the poker-based vignettes (C and D). The nature of vignettes A and B is such that they really must be played in a computer for them to make much sense. The reason for using a simulation, of course, was to maintain control on the data and avoid noise from the environment. Given that proof of concept of the learning of transitions was the main objective of this paper, we believe that this is justified. However, the question on how one would apply this approach when observing an actor in the real world arises. Our response is that, in an ideal world, our approach could be used in such a situation as long as the features of the actor's actions could be extracted from the observations logged by some front-end process. For example, in poker, the motion of throwing down the card played signals a player's move. The front-end process would have to interpret this move and then focus on the card played to identify it. Alternatively, folding is signaled by laying down all cards and pushing them away from the player. Once this information is fed to our learning system, it would see no difference from having observed a simulation. However, the envisioned front-end process would be quite complex and beyond the scope of this research, at least for the poker application.

More generally, the feasibility of building an adequate front-end process to extract the features would depend on the application, i.e., the type of task being observed and learned. In the case of a physical task or activity where only the location, direction, and speed of a person or a vehicle become important, then a Global Positioning System transmitter that identifies these data to an observer may be sufficient to learn that actor's or vehicle's behavior. This was shown by Fernlund *et al.* [5], albeit using a different approach to learning from observation. However, applications that heavily depend on gestures or human gesticulated motions (such as throwing down cards) may require highly complex front ends to permit their use in learning from observation and would thereby be more limited in their application.

The second question that arises is whether this approach would work in continuous games or tasks that are not turn based. Clearly, turn-based games provide a natural cue for the context to potentially change. Such would not be the case in many continuous tasks such as controlling a vehicle (e.g., car and aircraft). While knowing the time of this (potential) context transition cue clearly simplifies the learning, we can project how such a system would work.

Our approach would be to look for an "interesting" action or event in the sequence of events being observed. Such an "interesting" event would indicate the triggers for the change in context, which is what we are trying to learn. The change in context itself could be identified by a TBI engine by identifying when a new template is used to describe the actions of the human actor. "Interesting" activities would include events, changes in behavior (e.g., slowing down and changing direction), the actions of others (e.g., an enemy fires upon the human actor), environmental occurrences (e.g., it starts to rain), or even geographical location (e.g., passing a landmark and reaching an exit in an interstate highway). Of course, the crux of this approach would be carefully defining the concept of "interesting," as well as determining how to identify all such events and actions just before and after the transition. Events and actions after the transition takes place could indicate anticipation by the human actor. While we did not address the issue of temporally continuous actions, it does remain an interesting subject of future research.

V. Testing and Evaluation of Concept

We subjected the prototype FAMTILE system to six *test scenarios* (TSs) to determine whether the concepts behind the prototype—the use of neural networks to learn context transition criteria from observation of a human performer—work as expected. As described in the previous section, we have developed four *vignettes* (A–D), each presenting the human test subjects with a different game in which to make decisions. We designed the six TSs to evaluate the effectiveness of our work. TSs 1 and 2 involve the first two vignettes, whereas TSs 3, 4, 5, and 6 involve the poker vignettes (C and D).

For this evaluation, four human test subjects (denoted here as *Alpha*, *Bravo*, *Charlie*, and *Delta*) are used. Three subjects participated in each of the four vignettes, but they were different ones for the various vignettes. This was done to accommodate their varying availabilities. The subjects were selected from a pool of students in the laboratory that had some experience with poker. Three of the subjects were male (Alpha, Bravo, and Charlie), whereas subject Delta was a female. With regard to the poker vignettes, two of the three participating subjects (Alpha and Charlie) considered themselves to be of moderate to advanced skill, whereas subject Delta was a relative novice. These subjects were asked to install the vignettes on their computer and play the games while the simulation recorded each of their decision points.

Subjects Alpha, Bravo, and Delta participated in TSs 1 and 2. These scenarios correspond to vignettes A and B, respectively, and evaluate the ability of FAM to learn relatively simple behaviors exhibited by the test subject in these vignettes, without the TBI context identification feature. The basic objective of TSs 1 and 2 was to evaluate the ability of a standalone FAM to learn human-performed actions in a simple game before applying them to a more complex game. In TSs 1 and 2, atomic actions are represented by directional choices: either left, right, up, or down. These directions are also representative of the entire action space of the behavior, as no other actions are permitted within the maze. In vignettes A and B, all possible contexts that may provide motivation for each action are ignored during training. For instance, the motivation of going left because the goal state is in that direction is considered to be identical to the motivation of going left simply because that is the best alternative. Because of this, contexts behind the selection of particular moves by the test subjects were not considered in these two testing scenarios. We should note, however, that contexts still exist on the part of the agent that moves in the simulation. It is just that they are not considered in the training.

In TSs 3 and 4, subjects Alpha, Charlie, and Delta performed the more complex activities related to vignettes C and D, respectively: participating in hands of Texas Hold'em. The objective of TSs 3 and 4 was to evaluate the ability of a standalone FAM system to learn the actions and play them back in a simulated game, regardless of the underlying contexts. The learning poker agent merely learned to map the game conditions (the environment) to the actions taken by the test subjects. Comparison of the results of TSs 3 and 4 later to those of TSs 5 and 6 would, furthermore, provide an indication of the value of learning to predict the underlying contexts rather than merely the actions. Vignettes C and D involve reasoning about several observations, where each may have a significant impact on the subject's eventual decision. Furthermore, each action taken by the subject may be the result of complex motivations, as would be appropriately defined in a context. For instance, a raise or a bet resulting from the action prescribed in one context may be caused by a different reason than it would in another context. TSs 3 and 4, however, intentionally ignore this fact. When a player makes an action, it is presented to FAM as that action, regardless of any context that may be behind it. Because of this, these tests mirror those of TSs 1 and 2, but with significantly more complex behaviors.

TSs 5 and 6 also employ vignettes C and D, respectively, and were executed by subjects Alpha, Charlie, and Delta. By contrast, TSs 5 and 6 consider the context of each subject action prior to creating a training pattern for the neural network. Before running TSs 5 and 6, a set of contexts was developed for both vignettes C and D in an effort to capture all possible motivations for each action. During training, the subject's action at each decision point is first examined by a TBI engine to infer a context for that point. In TS 5, vignette C is reused, and FAMTILE attempts to learn subject actions just as FAM attempted to do in TS 3. It is hypothesized that the representation of the subjects' actions as inferred contexts can help the network to more effectively make finer clusters representing more closely related patterns, thereby increasing the predictive accuracy of the system. For the FAM within FAMTILE, just as in TSs 3 and 4, the actions of the observed human performer (the subject) are presented as output patterns, regardless of the motivation behind the action.

A. *Evaluation Procedure*

The evaluation of the FAM learning process for TSs 3 and 4 was done as presented here.

- The entire observation sequence gathered from subject i is used to generate a set of training patterns—no validation set is generated.
- FAM is trained with a set of patterns and learns a mapping between observation and action.
- FAM replaces the test subject and is presented with various decision points as the game progresses.
- For each decision cue presented by the simulation, FAM predicts an action based on what it learned.
- That action is then executed in the simulation, and the vignette continues.
- The overall performance of both subject i and FAM is compared based on the metrics collected throughout the execution of the scenario.

When separately testing FAM (TSs 3 and 4), the network is trained with the subject's action being presented at its output. For FAMTILE (TSs 5 and 6), the actions of the subject are first translated to an inferred context (by the TBI) for each decision point, and a representation of that context is presented to the FAM network within FAMTILE. After the training of each system was completed, the simulation was run again. This time, each decision cue was presented to the newly trained poker agent. Based on its knowledge, then, the poker agent running FAMTILE predicts a context, and the actions associated with that context were executed. In contrast, the standalone FAM produces only a predicted action. Six steps for testing the full FAMTILE system are given here.

1) The entire observation sequence gathered from subject i is used to generate a set of training patterns. Both the training and validation sets are taken from these observations.
2) FAMTILE is trained with the complete set of patterns and generates a mapping between the observation and the context.
3) FAMTILE takes the place of the subject within the simulation and executes the appropriate vignette.
4) For each decision cue presented by the simulation, FAMTILE predicts a context.
5) The identified context provides an appropriate action that is then executed. The vignette continues.
6) The overall behaviors of both subject i and FAMTILE are compared based on the metrics collected throughout the execution of the vignette.

For each scenario, the following FAM parameters were held constant:

- $\varepsilon = 0.00001$;
- $\beta_a = \beta_b = 1$;
- $\rho_b = 1$.

TABLE III
SUMMARIZED RESULTS FOR SCENARIO 1

Subject	$\overline{\rho}_a$	$\overline{\rho}_{a_{test}}$	$\overline{\mu}$	$\overline{\sigma}$
Alpha	0.6	0	94.7	2.38
Bravo	0.8	0	87.3	3.27
Delta	0.8	0	80.6	3.76

The only parameter that was modified during the testing phase was the baseline vigilance $\overline{\rho}_a$. This parameter has a direct effect on the granularity of the clusters generated in the ART_a module. These clusters represent groups of input patterns presented to ART_a, where each pattern maps to the same output pattern (either an action as in TSs 1, 2, 3 and 4, or a context as in TSs 5 and 6) and is closely matched with respect to its individual field. The baseline vigilance parameter $\overline{\rho}_a$ affects this granularity by raising the vigilance parameter, which is responsible for rejecting the addition of new input patterns to a certain cluster if it fails to match a certain criteria. This change ultimately increases the number of input pattern clusters created in ART_a by decreasing their overall size (and inclusiveness). This effect is quantitatively illustrated in the succeeding sections.

B. *TS 1 Results*

Essentially, the task for FAM in this TS is to create a mapping between the maze topology and a predicted direction for the test subject facing that situation: either left, right, up, or down.

The intent of vignette A is to create an environment where the actions of the subject are closely tied to the primary goals of the behavior. In this vignette, the subject makes only a single move in response to being told where and how far away the goal position is. Each atomic move, therefore, is made in direct accordance with the objective of reaching the goal. In the next few vignettes, the behavior required becomes increasingly complex, and the relationship between the atomic actions required by the subject consequently become less dependent on the overall objective and more dependent on the context in which the subject is operating.

The testing proceeded in five steps.

1) Randomize the order of the 1000 training points.
2) Choose 900 of the 1000 points at random to train the neural network; use the final 100 points for the validation set.
3) Train the neural network using the 900 chosen training points.
4) Test the neural network using the remaining 100 points.
5) Record the number of correct predictions made out of 100 testing patterns.

Table III displays the results for each subject, including the sample mean predictive accuracy μ and standard deviation $\overline{\sigma}$. A 2-tailed t-test was used on each set of data to validate that the computed sample mean $\overline{\mu}$ for each subject approaches the actual mean μ. Using an α value of 0.01, the test computed a 99% confidence interval for the actual mean.

As expected, FAM is able to successfully learn the movement patterns for each of the three subjects. Success, here, is defined as better than random. A random guess at the subject's action for vignette A would yield, on average, 25% predictive accuracy (because there are four possible actions). As a qualitative

TABLE IV
SUMMARIZED RESULTS FOR TS 2

	$\overline{\rho}_a$	$\overline{\rho}_{a_{test}}$	$\overline{\mu}$	$\overline{\sigma}$
Alpha	0.8	0	92.5	2.63
Bravo	0.8	0	84.5	3.42
Delta	0.7	0	85.6	3.31

TABLE V
AVERAGE PREDICTIVE ACCURACY FOR TS 3 USING OPTIMAL $\overline{\rho}_a$ VALUES

	$\overline{\mu}$	$\overline{\sigma}$
Alpha	75.04	4.20
Delta	68.54	4.46
Charlie	75.56	3.68

comparison, consider the accuracies achieved by each subject. For subject Alpha, the network was able to predict, on average, almost 95 of the 100 testing patterns. Even for the worst cased subject (TS 3), FAM was able to predict nearly 81% of the testing patterns.

The purpose is for these results to serve as a baseline to evaluate FAM (and ultimately FAMTILE) and examine how this notion of context affects their predictive accuracy.

C. TS 2 Results

TS 2 was executed in the same manner as TS 1, and the same three subjects were used. Within this scenario, each subject makes consecutive moves within a 10×10 maze, with the board and goal positions resetting each time the subject reaches the goal. The scenario ends when the subject has generated 1000 training points—each training point represents a specific maze state and the action the subject makes in response to that state. Those points were used to train and evaluate the neural network. Table IV displays the results of the 1000 run sets for each subject.

In this scenario, FAM was able to adequately learn the movement patterns for each of the three subjects. Furthermore, the predictive accuracy significantly varied across subjects, just as it had in scenario 1. FAM achieved a predictive accuracy of nearly 93 of 100 for subject Alpha versus 84.5 and 85.6 for the other two.

D. TS 3 Results

In vignette C, each of three test subjects is placed at a simulated Texas Hold'em game with seven computer-generated opponents. As expected, the predictive accuracy of FAM significantly degraded when tested using vignette C as a result of the greater complexity of the problem. By the numbers, the network achieved best-case predictive accuracies of 75.0, 68.5, and 75.6 for each player versus 92.5, 84.5, and 85.6 for TS 2, respectively (see Table V).

Comparing the predictive accuracies of FAM on these two subjects for TSs 2 and 3, there is a 17.5% decrease in predictive accuracy for subject Alpha and a 17.1% decrease for subject Delta. This is a sharp contrast to the statistically insignificant performance difference between TSs 2 and 1, where the network's predictive accuracy changed to 2.2% and 2.8% for subjects Alpha and Delta, respectively. These results confirm that the poker environment of vignette C is much more complex and therefore harder for FAM to learn versus that of the simpler maze vignettes. What this means in terms of the network itself is that FAM had a more difficult time effectively creating clusters with similar data points that mapped to the output patterns representing correct predictions of the subject's action.

An interesting result of this test was the sharp contrast in the predictive accuracy of FAM for subject Delta versus the other two subjects. As previously reported, FAM was only able to predict 68.54% of subject Delta's actions versus 75.04 and 75.56% for the other two subjects. One hypothesis as to this discrepancy is the difference in skill between subject Delta and subjects Alpha and Charlie. In Texas Hold'em, proper play before the flop is both the easiest piece of strategy to learn and the most crucial [69]. Strategy after this round becomes much more complex because of the explosion of information present with community cards on the board. Because of this, Limit Hold'em play before the flop round of betting tends to be somewhat mechanical among experienced players. This is supported by the data on subjects Alpha and Charlie, who shared similar experiences and read much of the same literature. Subject Delta (the novice player as previously described), on the other hand, has much less experience; thus, her play is likely to be more erratic and, therefore, less predictable. However, a similar drop-off between subject Delta versus subjects Alpha and Charlie is present in the results reported in scenario 1 (although not in scenario 2). Because of this, another hypothesis for the change in predictive accuracies is the level of attention Delta paid to the exercise for vignettes A and C. Since the participants did not execute each vignette in sequence (and was not monitored during the exercises), it is possible that Delta simply was not paying full attention during the exercises. This hypothesis is bolstered by the more reasonable results of scenario 2, where the decision points were much more straightforward (navigating an entire maze versus simply making a single decision of direction).

E. TS 4 Results

In TS 4, the predictive accuracies for FAM were collected and analyzed for vignette D. Just as vignette C, this vignette is set at the poker table with seven computer-generated agents playing against the subject in games of Texas Hold'em. Here, however, the subject's decision points are not limited to the first round of action. Instead, a series of entire hands are carried out to their completion: if a subject folds, a new hand is dealt; if a subject raises, the opponents accordingly react to that raise; a flop, turn, and river are dealt; and betting rounds follow just as in an actual hand. The subject is also given a stack of 100 "chips" that is maintained throughout the vignette. In this fourth and final evaluation of the FAM, we continue to examine its ability to learn subject actions as a function of his cards, his position at the table, and the betting action.

Once again, the increase in complexity of vignette D compared to vignette C resulted in further erosion in the FAM's predictive accuracy. The best-case accuracies of 55.32, 58.95, and 58.12 (see Table VI) are an average of more than 20% worse than those of scenario 3, which is nearly twice the decrease observed between vignette C and the maze scenarios.

TABLE VI
AVERAGE PREDICTIVE ACCURACY FOR TS 4 USING OPTIMAL $\overline{\rho}_a$ VALUES

	$\overline{\mu}$	$\overline{\sigma}$
Alpha	55.32	5.24
Charlie	58.95	4.47
Delta	58.12	2.91

TABLE VII
SUMMARIZED RESULTS FOR SCENARIOS 3 AND 5

	$\overline{\mu}_1$	$\overline{\mu}_2$	$\overline{\mu}_1 - \overline{\mu}_2$	**99%CI**	**p-value**
Alpha	75.63	75.40	0.224	(-0.228,0.676)	0.201
Delta	68.92	68.55	0.372	(-0.135,0.879)	0.059
Charlie	75.37	75.56	-0.187	(-0.666,0.292)	0.315

It was observed in TS 3 that FAM significantly performed worse on Delta than on the other two experts. Furthermore, it was noted that Delta had several years fewer experience than the other two, which possibly affected the predictability and consistency of the actions.

The complexity of this scenario, however, seems to have neutralized this effect. In fact, FAM was slightly more effective in the best case at predicting expert Delta's actions than those of the other two experts. As it turns out, Charlie (who did not participate in vignette C or the maze vignettes) had comparable experience as expert Alpha.

F. TS 5 Results

The objective for TS 5 is to evaluate FAMTILE's ability to predict both the subject's inferred active context and his resultant action. Vignette C is used for this TS, which is the same one used to evaluate FAM in testing scenario 3. Because of this, the results of TS 3 serve as a baseline performance metric for the results achieved here. Unlike FAM, however, FAMTILE instead attempts to predict the subject's inferred active context. In order to make a comparison between FAM and FAMTILE, the predicted contexts of FAMTILE must then be converted to a predicted action for the subject, using the contents of the predefined context template. Because FAM does not make context predictions, this determination is necessary to compare the predictive accuracies of the two learning systems. The results of scenario 5 are presented in Table VII (represented by $\overline{\mu}_1$), along with those from scenario 3 (represented by $\overline{\mu}_2$), using 900 training patterns.

There are several interesting things to note from these results. In terms of the primary objectives of this research, the numbers in the third column are the most important—how well does FAMTILE predict the inferred context of the subject? As Table VII illustrates, these predictive accuracies of the subject's action for FAM and FAMTILE are nearly identical for each batch of runs and each subject. In the best case, for subject Alpha with 900 training patterns, FAMTILE outperformed FAM with an average of 75.63 correct predictions versus 75.04 for FAM. In the worst case, for subject Delta, FAM narrowly outperformed FAMTILE with an average of 75.56 correct predictions versus 75.37 for FAMTILE. However, neither of these margins is statistically significant.

TABLE VIII
AVERAGE CONTEXT-PREDICTIVE ACCURACY FOR TS 5

	$\overline{\mu}$ **(context)**	$\overline{\sigma}$
Alpha	67.71	4.04
Delta	59.98	4.81
Charlie	66.26	5.17

TABLE IX
SUMMARIZED RESULTS FOR SCENARIOS 4 AND 6

	$\overline{\mu}_1$	$\overline{\mu}_2$	$\overline{\mu}_1 - \overline{\mu}_2$	**99%CI**	**p-value**
Alpha	60.25 (75.63)	58.22 (75.40)	2.30	(1.253,3.347)	0.778
Delta	60.14 (68.92)	60.18 (68.55)	-0.04	(-0.460,0.380)	0.006
Charlie	54.07 (75.37)	55.32 (75.56)	-1.25	(-2.38,-0.120)	0.572

In addition, FAMTILE is able to accurately predict the subject's active context an average of 67.71, 59.98, and 66.26 times for each of the three subjects observed, respectively, at optimum values for $\overline{\rho}_a$ (see Table VIII). Comparing these accuracies with those of FAM for predicting subject actions, we note that FAMTILE is an average of only 11.52% less effective at predicting contexts than FAM is at predicting actions.

The fact that FAMTILE is able to generate a competitive degree of context-predicting accuracy *without* disrupting the ability of FAM is significant. In effect, therefore, we have created a system that adds the ability to predict context transitions to a neural network without significantly affecting its ability to predict simple actions.

G. TS 6 Results

In scenario 6, predictive accuracies for FAMTILE are collected and analyzed for vignette D as they were for FAM in scenario 4. Table IX summarizes the results of a 2-tailed t-test on the best-case predictive accuracy means achieved in scenarios 4 ($\overline{\mu}_2$) and 6 ($\overline{\mu}_1$) for each subject. In the table, the values from scenarios 3 and 5 are annotated in parentheses.

The predictive accuracy of FAMTILE for predicting the subject's inferred context also considerably decreased from the values achieved in scenario 5. Whereas FAMTILE predicted contexts at rates of 67.71, 59.98, and 66.26 for vignette C, those accuracies dropped by an average of more than 28% across the two subjects who then also participated in vignette D. One significant reason for this was the increase in the number of contexts. This number doubled from 12 to 24 contexts for vignette D, because two new actions needed to be accounted for (i.e., bet and check), along with the representation of contexts potentially present after the preflop round of betting. Note that, with 24 contexts, a random guess of the inferred active context could be expected to be correct slightly more than 4% of the time, which is ten times less than the accuracy achieved by FAMTILE.

Furthermore, vignette D requires the player to reason about entirely new and more complex situations than those faced in vignette C. In addition to his/her hole cards, the player must also consider not only the community cards but also the action of previous betting rounds and the possible responses of each opponent in response to a particular action.

VI. Conclusion and Lessons Learned

Based on the results tabulated in the previous section, it is concluded that FAMTILE is an adequate technique for learning high-level behaviors and offers several promising characteristics that can be exploited in future research. Because it is able to learn low-level contexts from human actors without adversely affecting the clustering ability of FAM, we feel that the FAMTILE system provides a significant tool for learning in systems where it is desirable to gain a perspective of *why* the human actor is doing what he/she is doing.

The results of the two maze scenarios provide a good indication as to FAM's ability to predict human responses to an observation. In TS 1, the network is able to correctly predict a subject's movement at an average of 86% on the validation set, achieving nearly a 95% average for one of the three subjects. This scenario included input training patterns with 27 fields and four possible output patterns. The second maze TS expanded the subject's viewing range, more than tripling the number of input-pattern fields to 88 (92 if the subject's previous action was recorded and considered). Nevertheless, FAM is able to predict 85% of the validation set for the three subjects, increasing to nearly 87% when the subject's previous action is considered.

While these are impressive numbers for predicting three different subject's actions, they only speak to the successes of FAM and do not address the capabilities of FAMTILE. These scenarios were executed and reported, for the most part, to justify the use of FAM for doing the low-level learning task. Had these evaluations been a failure, a different learning system would have had to be selected—one that performed better at predicting actions within these training scenarios.

As described in Sections IV and V, FAMTILE requires the use of a completely separate TBI module that encodes *a priori* knowledge about the scenario within its context templates, whereas FAM itself requires no such input. FAMTILE fails to produce a worthwhile increase in predictive performance, therefore negating our hypothesis. A separate set of tests was run to evaluate FAMTILE's ability to correctly predict the inferred expert context for each decision point. While these tests resulted in lower predictive accuracies—certainly expected because the neural network must choose between 12 possible output patterns, instead of only three, when predicting actions—the results were promising. Using 900 training patterns, FAMTILE is able to correctly predict an average of 64.77 contexts out of a possible 100 (64.77%) across the three experts. As reported in Section VI, FAMTILE's predictive accuracy for contexts is only around 11% worse than its accuracy for actions. This accuracy is achieved, furthermore, without affecting the accuracy of the network in predicting the expert's overall action. What this means, then, is that FAMTILE can provide a significant advantage over other supervised learning algorithms in situations where the identification of expert context provides more important or additionally worthwhile information versus simply being able to predict low-level action. In a more robust poker simulation, for example, the ability of FAMTILE to identify context could drive additional behaviors, aside from the simple game action, such as additional "table talk" to project a strong image while bluffing, voice intonation, etc. Generally, we feel that the FAMTILE system is most useful for learning tasks where three conditions hold.

1) The behavior satisfies the characteristics of high-level tactical behavior, as defined in Section I.
2) The user is interested in creating models of the expert's behavior and is more interested in his resultant intentions and motivations than the actions observed at the lowest level.
3) The expert's ultimate action is more closely tied to his low-level behavior than to the raw observation presented at each decision point.

This difference in difficulty between the maze and the poker vignettes seemed to create a good set of conditions for evaluating both FAM and FAMTILE. The first human-prediction task (the maze) was found to be relatively easy yet reflected some variability among the three subjects observed. The second two TSs introduced the poker scenario. These vignettes introduce a learning challenge that, while containing a comparable number of input-pattern fields and output possibilities, proved to be a more difficult task for both FAM and FAMTILE.

FAMTILE requires the use of a separate TBI module that encodes *a priori* knowledge about the scenario within its context templates, whereas FAM itself requires no such input. FAMTILE fails to produce a worthwhile increase in predictive performance.

The central assumption made for this research was that high-level behavior can be represented by a sequence of lower level behaviors that can be modeled by CxBR contexts. However, the trick then becomes defining and partitioning each context of a behavior in such a manner that they are truly atomic and identifiable, independent of the specific subject being observed. For example, consider the *RaiseWithStrongButVulnerableHand* context used in vignette D. This context was modeled to represent cases where the subject believes not only that he has the best hand at the moment but also that his opponents can easily draw cards to beat him.

This context raises an interesting question: What if the subject does not actually recognize this? Obviously, then, the templates must be defined such that this context is not inferred. However, what if there are no contexts that accurately represent the low-level motivation and behavior of the human subject?

High-level behaviors whose specifics are heavily dependent on human preference and expertise are equally difficult to represent. While a significant amount of *a priori* knowledge was encoded into the context templates used for scenarios 3 and 4, that knowledge does not represent the full range of motivations and contexts that constitute the entire task of playing Hold'em Poker. This is because these contexts are so dependent on the tendencies of the individual subject. Some players may employ poor strategies, for instance, that are not represented as a high-level context template. These absences can ultimately reduce the predictive accuracy of the FAMTILE system.

However, that is not to say that these assumptions serve only to doom the chances of success for our approach. On the contrary, these assumptions provide a means for motivating the directions that research in human behavior representation should progress. If we choose to learn a task where the modeling architecture, subject tendencies, and context topologies are all known, it is likely that the task modeled is too simple and not worth modeling. Texas Hold'em Poker, on the other hand, is a highly complex game, and the number of techniques, strategies, and styles documented and used by advanced players suggest that the game is as much of an art as it is a science.

REFERENCES

[1] *Webster's New World Dictionary of the American Language*, 1960, Cleveland, OH: World.
[2] A. J. Gonzalez and R. Ahlers, "Context-based representation of intelligent behavior in training simulations," *Trans. Soc. Comput. Simul. Int.*, vol. 15, no. 4, pp. 153–166, Mar. 1998.
[3] A. J. Gonzalez, J. Castro, and W. J. Gerber, "Automating the acquisition of tactical knowledge for military missions," *J. Defense Model. Simul.*, vol. 3, no. 1, pp. 145–160, Spring 2006.
[4] T. A. Sidani and A. J. Gonzalez, "A framework for learning implicit expert knowledge through observation," *Trans. Soc. Comput. Simul. Int.*, vol. 17, pp. 54–72, 2000.
[5] H. Fernlund, A. J. Gonzalez, M. Georgiopoulos, and R. F. DeMara, "Learning tactical human behavior through observation of human performance," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 1, pp. 128–140, Feb. 2006.
[6] R. M. Jones, J. E. Laird, P. E. Nielsen, K. J. Coulter, P. G. Kenny, and F. Koss, "Automated intelligent pilots for combat flight simulation," *AI Mag.*, vol. 20, no. 1, pp. 27–41, 1999.
[7] R. M. Jones and J. E. Laird, "Constraints on the design of a high-level model of cognition," Univ. Michigan, Ann Arbor, MI, 1996. Res. Rep.
[8] D. J. Patterson, L. Liao, D. Fox, and H. Kautz, "Inferring high-level behavior from low-level sensors," Univ. Washington, Seattle, WA, 2003. Res. Rep.
[9] R. Khardon, "Learning to take actions," Dept. Comput. Sci., Univ. Edinburgh, Edinburgh, U.K., Jun. 12, 1998. Tech. Rep.
[10] M. van Lent and J. E. Laird, "Learning hierarchical performance knowledge by observation," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 229–238.
[11] C. E. Shannon, "Programming a computer for playing chess," *Philos. Mag.*, vol. 41, no. 314, pp. 256–275, 1950.
[12] A. M. Turing, "Digital computers applied to games," in *Faster Than Thought: A Symposium on Digital Computing Machines*, B. V. Bowden, Ed. London, U.K.: Pitman, 1953, ch. 25, pp. 286–310.
[13] A. Newell, J. C. Shaw, and H. A. Simon, "Chess-playing programs and the problem of complexity," *IBM J. Res. Develop.*, vol. 2, no. 4, pp. 320–335, 1958.
[14] N. Charness, "Components of skill in bridge," *Can. J. Psychol.*, vol. 33, no. 1, pp. 1–16, Mar. 1979.
[15] N. Charness, "Component processes in bridge bidding and novel problem-solving tasks," *Can. J. Psychol.*, vol. 41, no. 2, pp. 223–243, Jun. 1987.
[16] N. Charness, "The impact of chess research on cognitive science," *Psychol. Res.*, vol. 54, no. 1, pp. 4–9, Mar. 1992.
[17] T. Jastrzembski, N. Charness, and C. Vasyukova, "Expertise and age effects on knowledge activation in chess," *Psychol. Aging*, vol. 21, no. 2, pp. 401–405, Jun. 2006.
[18] F.-H. Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton, NJ: Princeton Univ. Press, 2002.
[19] M. Newborn and D. Kopec, "The twentieth annual ACM North America computer chess championship," *Commun. ACM*, vol. 33, no. 7, pp. 92–105, Jul. 1990.
[20] D. Coursey, "Deep Thought is a chess-playing computer," *MIS Week*, vol. 11, no. 14, p. 33, Apr. 2, 1990.
[21] A. L. Zobrist, "Feature extraction and representation for pattern recognition and the game of GO," Ph.D. dissertation, Univ. Wisconsin, Madison, WI, 1970.
[22] J. L. Ryder, "Heuristic analysis of large trees as generated in the game of Go," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1971.
[23] A. Kierulf and J. Nievergelt, "Computer GO: A smart board and its applications," in *Go Worlds*, vol. 42. Tokyo, Japan: Ishi Press, Winter 1986.
[24] A. Kierulf, K. Chen, and J. Nievergelt, "Smart game board and go explorer," *Commun. ACM*, vol. 33, no. 2, pp. 152–167, Feb. 1990.
[25] B. Wilcox, "Reflections on building two GO programs," *SIGART Newsl.*, vol. 94, pp. 29–43, Oct. 1985.
[26] A. L. Samuel, "Some studies in machine learning using the game of checkers," in *Computers and Thought*, E. A. Feigenbaum and J. Feldman, Eds. New York: McGraw-Hill, pp. 71–105.
[27] D. A. Waterman, "Generalization learning techniques for automating the learning of heuristics," *Artif. Intell.*, vol. 1, no. 1/2, pp. 121–170, 1970.
[28] R. S. Michalski, J. G. Carbonel, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*. Los Altos, CA: Morgan Kaufmann, 1986.
[29] M. X. Zhou and S. Ma, "Toward applying machine learning to design rule acquisition for automated graphics generation," IBM T. J. Watson Res. Center, Hawthorne, NY, 2000. Res. Rep.
[30] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to fly," in *Proc. 9th Int. Conf. Mach. Learn.*, Aberdeen, U.K., 1992, pp. 335–339.
[31] R. Camacho, "Inducing models of human control skills," in *Proc. 10th ECML*, Chemnitz, Germany, 1998, pp. 107–118.
[32] A. Isaac and C. Sammut, "Goal-directed learning to fly," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, DC, 2003, pp. 258–265.
[33] A. E. Henninger, "Neural network based movement models to improve the predictive utility of entity state synchronization methods for distributed simulations," Ph.D. dissertation, School Electr. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, 2001.
[34] W. J. Gerber, "Real-time synchronization of behavioral models with human performance in a simulation," Ph.D. dissertation, School Electr. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, 2001.
[35] M. J. Johnson, M. McGinnis, M. Mollaghasemi, and T. Damarly, "Methodology for human decision making using fuzzy ARTMAP neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Honolulu, HI, 2002, pp. 2668–2673.
[36] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, Sep. 1992.
[37] M. van Lent and J. E. Laird, "Learning procedural knowledge through observation," in *Proc. 1st Int. Conf. K-CAP*, Victoria, BC, Canada, 2001, pp. 179–186.
[38] T. Konik and J. E. Laird, "Learning goal hierarchies from structured observations and expert annotations," in *Proc. 14th Int. Conf. Inductive Logic Program.*, 2004, pp. 198–215.
[39] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, Jun. 1999.
[40] S. Walczak and P. Fishwick, "A quantitative analysis of pattern production and its relationship to expert performance," *J. Exp. Theor. Artif. Intell.*, vol. 9, no. 1, pp. 83–101, Jan. 1997.
[41] J. E. Laird, P. Rosenbloom, and A. Newell, *Universal Subgoaling and Chunking*. Norwell, MA: Kluwer, 1986.
[42] D. Pomerlau, C. Thorpe, D. Longer, J. K. Rosenblatt, and R. Sukthankar, "AVCS research at Carnegie Mellon University," in *Proc. Intell. Vehicle Highway Syst., Amer. Annu. Meeting*, 1994, pp. 257–262.
[43] D. Bentivegna and C. Atkeson, "Learning from observation using primitives," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, May 21–26, 2001, pp. 1988–1993.
[44] A. Moukas and G. Hayes, "Synthetic robotic language acquisition by observation," in *Proc. 4th Int. Conf. Simul. Adapt. Behav.*, 1996, pp. 568–579.
[45] B. Yang and H. Asada, "Hybrid linguistic/numeric control of deburring robots based on human skills," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1992, vol. 2, pp. 1467–1474.
[46] D. Floreano and F. Mondada, "Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot," in *Proc. Conf. Simul. Adapt. Behav.*, 1994, pp. 421–430.
[47] A. Pentland and A. Liu, "Toward augmented control systems," in *Proc. Intell. Vehicles*, 1995, vol. 1, pp. 350–355.
[48] L. J. Fogel, W. Porto, and M. Owen, "An intelligently interactive non-rule-based computer generated force," in *Proc. 6th Conf. Comput. Generated Forces Behavior Representation*, Orlando, FL, Jul. 23–25, 1996, pp. 265–271.
[49] J. D. Morrison, "Real time learning of doctrine and tactics using neural networks and combat simulations," *Mil. Oper. Res.*, vol. 2, no. 3, pp. 45–60, 1996.
[50] M. Crowe, "The application of artificial neural systems to the training of air combat decision-making skills," in *Proc. 12th Inter-Service/Industry Training Syst. Conf.*, Orlando, FL, 1990, pp. 302–312.
[51] H. Friedrich, M. Kaiser, and R. Dillman, "What can robots learn from humans?," *Annu. Rev. Control*, vol. 20, pp. 167–172, 1996.
[52] M. Kaiser and R. Dillman, "Building elementary robot skills from human demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, 1996, pp. 2700–2705.
[53] S. Rajput, C. R. Karr, J. E. Cisneros, and R. J. Parsons, "Learning the selection of reactive behaviors," in *Proc. 6th Conf. Comput. Generated Forces Behavioral Representation*, Orlando, FL, Jul. 23–25, 1996, pp. 3–8.

[54] M. R. Hieb, G. Tecuci, and M. Pullen, "Training ModSAF command agent through demonstration," in *Proc. 6th Conf. Comput. Generated Forces Behavioral Representation*, Orlando, FL, Jun. 1996.

[55] C. G. Gingrich, D. R. Kuespert, and T. J. McAvoy, "Modeling human operators using neural networks," *ISA Trans.*, vol. 31, no. 3, pp. 81–90, 1992.

[56] G. E. Hovland, P. Sikka, and B. J. McCarragher, "Skill acquisition from human demonstration using a hidden Markov model," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1996, pp. 2706–2711.

[57] K. Kosuge, T. Fukuda, and H. Asada, "Acquisition of human skills for robotic systems," in *Proc. IEEE Intell. Symp. Intell. Control*, 1991, pp. 469–474.

[58] S. Lee and J. Chen, "Skill learning from observations," in *Proc. IEEE Conf. Robot. Autom.*, 1994, vol. 4, pp. 3245–3250.

[59] S. Lee and H. Chen, "Robot skill discovery based on observed data," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, 1996, pp. 2694–2699.

[60] A. Modjtahedzadeh and R. A. Hess, "A model of driver steering control behavior for use in assessing vehicle handling qualities," *Trans. ASME, J. Dyn. Syst. Meas. Control*, vol. 115, no. 3, pp. 456–464, 1993.

[61] E. Fix and H. G. Armstrong, "Neural network based human performance modeling," in *Proc. IEEE Nat. Aerosp. Electron. Conf.*, 1990, vol. 3, pp. 1162–1165.

[62] M. C. Nechyba and Y. Xu, "Learning and transfer of human real-time control strategies," *J. Adv. Comput. Intell.*, vol. 1, no. 2, pp. 137–154, 1997.

[63] M. C. Nechyba and Y. Xu, "Cascade neural networks with node-decoupled extended Kalman filtering," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, 1997, vol. 1, pp. 214–219.

[64] P. Drewes, "Automated student performance monitoring in training simulation," Ph.D. dissertation, Dept. Electr. Comput. Eng., Univ. Central Florida, Orlando, FL, 1997.

[65] M. Georgiopoulos and C. Christodoulou, *Applications of Neural Networks in Electromagnetics*. Norwood, MA: Artech House, 2001.

[66] B. S. Stensrud, "FAMTILE: An algorithm for learning high-level, tactical behavior from observation," Ph.D. dissertation, School Electr. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, 2005.

[67] D. Brunson, *Doyle Brunson's Super System*, 3rd ed. New York: Cardoza, 1979.

[68] D. Sklansky, *The Theory of Poker*, 3rd ed. Henderson, NV: Two Plus Two Publications, 1989.

[69] D. Sklansky and M. Malmouth, *Hold'em Poker for Advanced Players*. Las Vegas, NV: Creel, 2003. 21st Century Edition.

[70] D. Billings, D. Papp, J. Schaeffer, and D. Szafron, "Poker as a testbed for AI research," in *Proc. 12th Can. Conf. Artif. Intell.*, 1998, pp. 228–238.

Brian S. Stensrud (M'06) was born in Orlando, FL, on October 3, 1978. He received the B.S. degrees in computer engineering, electrical engineering, and mathematics (with honors) from the University of Florida, Gainesville, in 2001 and the M.S. and Ph.D. degrees in computer engineering from the University of Central Florida, Orlando, in 2003 and 2005, respectively.

He was a Research Assistant, under Dr. A. Gonzalez, with the University of Central Florida. Since September 2003, he has been with Soar Technology, Inc., Orlando, where he is currently a Research Scientist. His current research interests include knowledge-rich intelligent agents and asymmetric behavior generation.

Avelino J. Gonzalez (F'07) received the B.S. and M.S. degrees in electrical engineering from the University of Miami, Coral Gables, FL, in 1973 and 1974, respectively, and the Ph.D. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, in 1979.

He is currently a Professor with the Intelligent Systems Laboratory, School of Electrical and Computer Engineering, University of Central Florida, Orlando, specializing in artificial intelligence and simulation.