Context-driven Near-term Intention Recognition

Avelino J. Gonzalez
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
gonzalez@ucf.edu

Ronald F. DeMara
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
demara@mail.ucf.edu

William J. Gerber
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
bill.gerber@ieee.org

Michael Georgiopoulos
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
michaelg@mail.ucf.edu

Recognizing the intention of others in real time is a critical aspect of many human tasks. This article describes a technique for interpreting the near-term intention of an agent performing a task in real time by inferring the behavioral context of the observed agent. Equally significant, the knowledge used in this approach can be captured semi-automatically through observation of an agent performing tasks on a simulator in the context to be recognized. A hierarchical, template-based reasoning technique is used as the basis for intention recognition, where there is a one-to-one correspondence between templates and behavioral contexts or sub-contexts. In this approach, the total weight associated with each template is critical to the correct selection of a template that identifies the agent's current intention. A template's total weight is based on the contributions of individual weighted attributes describing the agent's state and its surrounding environment. The investigation described develops and implements a novel means of learning these weight assignments by observing actual human performance. It accomplishes this using back-propagation neural networks and fuzzy sets. This permits early discrimination between different pre-categorized behavioral contexts/sub-contexts on the human-controlled agent such as a military or passenger vehicle. We describe an application of this concept and the experimentation to determine the viability of this approach.

Keywords: DIS, network bandwidth

1. Introduction and Background

There often is a need to determine the intention of others before making decisions. A driver making a left turn typically uses the left turn signal to advise other drivers of her intention, thereby preventing dangerous actions by other motorists. Strategic team games virtually require that both teams be able to predict the intention of the opposition when designing plays.

In warfare, law enforcement and anti-terrorist activities, however, determining the intent of the enemy often becomes a life and death issue. Yet, like in team sports, it is quite unrealistic to directly ask the enemy about their intent. It must be inferred by unobtrusive observation, and it must be done in real time.

Our work presented here is based on the assumption that one's near-term intentions typically are based on a contextualized behavior — a set of actions and procedures humans perform while in a specific situation. Likewise, contextualized behaviors and their associated actions are used to control an agent while in a particular situation. We assert that once an agent's near-term intentions have been identified by the observer, future (near-term) actions by the observed agent can be predicted relatively easily. For example, if the observer sees that a motorist agent intends to turn left (left turn signal on and approaching an intersection), he can predict its near-term movement very accurately, and thereby act accordingly.

We base our approach on inferring the *behavioral context* of an observed agent whose intention we wish to discover. Behavioral contexts are defined in the *Context-based Reasoning* (CxBR) modeling paradigm for human tactical behavior representation. See [1] for details on CxBR. If the observer can infer the context in which an agent is operating, then prediction of the agent's actions would follow relatively easily. In CxBR, the context currently controlling an agent (i.e., the *active*

JDMS, Volume 1, Issue 3, August 2004 Page 153–170

context) contains the functionality to allow the agent to successfully "navigate" through the current situation. This approach relies on the fact that, normally, only a limited number of contexts could be realistically used by the agent in that situation. Furthermore, we assume that there is no desire on the part of the observed agent to disguise its intent to mislead the observer. However, the approach could be modified to account for purposeful deception in future work.

1.1 Previous Work

Previous research related to intention and plan recognition is indeed extensive. Schmidt and his colleagues pioneered the field of plan recognition with their seminal paper in 1978 [2]. In their BELIEVER system, they posed psychological theories for understanding how humans view and infer the plans of others by observing their actions.

Other research in intention recognition investigated the use of grammar parsing methodology to recognize behavior as matching previously defined sequences of events [3,4] while others investigated neural networks to do the same [5]. Wang & Arbib's model [4], however, required that the complete pattern be presented before the pattern recognition would occur. This would not be useful for early recognition of actions to predict future behavior in real time.

In a different vein, a few investigators ran competing models of the expected behaviors in parallel, either as Kalman filters [6] or linear models [7]. They then observed which model best tracked the observations and used that model of behavior to represent the current behavior. The results of Liu & Pentland [6] were mixed, with typical success rates of between 40% and 70% in real-time tests, depending on the circumstances. In reports of later experiments, Pentland & Liu [8] used a hierarchy of Markov dynamic models to represent long-time-scale driver behavior and fine-grain behavior. The Markov dynamic models using patterns of acceleration and heading produced a 95 ± 3% recognition accuracy two seconds after the command.

In one case [7], a model used for system control was adapted in real time to further reduce control errors. One unique research effort [9] used supervised neural network learning of processed inputs to directly discriminate between three distinct behaviors, showing the feasibility for automated machine learning of behavior at some level. Their results showed a maximum of 95% accuracy, but only one of the different neural networks tried could recognize more than 85% of the test cases. Furthermore, there is some question as to whether the recognition was conducted post facto, thereby precluding the advantage of predicting future behaviors by the agent. Another investigation [10] hints that the examples of behavior could be clustered using automated self-organization.

Strohal and Onken [11] describe the Crew Assistant Military Aircraft (CAMA) system, a knowledge-based assistant to enhance situation awareness for crews of future military transport aircraft. To accomplish that goal, CAMA had to assess the situation on its own, including the crew's intent. It was designed to infer the crew's intentions, permitting the system to anticipate the need for assistance without a request by the pilot. They used neuro-fuzzy techniques, but required a human to translate the resulting learned knowledge into rules usable by CAMA.

Plan recognition for human-computer collaboration is described by Lesh, et al., [12]. Plan recognition, as they define it, is "… the process of inferring intentions from action." Their work exploits the properties of the collaborative setting to make plan recognition practical. These properties are the focus of attention, partially elaborated hierarchical plans, and the possibility of asking for clarification.

Most recently, research in plan recognition has taken several different directions. The most popular of these involves developing logic theories to provide an algebra through which to reason about plans from observed agent actions. Wobke [13] presents an approach built upon Kautz's keyhole plan recognition work. He presents two approaches, one monotonic and one non-monotonic. The latter of these neglects Kautz's simplifying assumption of equal relevance for all competing plans. Wobke bases his approaches on defining a "... hierarchy of plan schemas." Jiang and Ma [14] introduce plan knowledge graphs, along with a new formalism, to simplify the process of plan recognition. They claim to reduce the plan recognition problem to a graph search with their approach, and obtain the same results as Kautz.

Patterson, et al. [15], address the problem of inferring high-level intentions from low-level sensors. They use Bayesian Nets to predict the position of a traveler in an urban setting, using auto, bus, and foot travel as the means of locomotion. They report high levels of accuracy in their predictions. Computer vision also has addressed the problem of plan recognition, albeit in different ways. Intille and Bobick [16] use Bayesian networks and model-based object recognition to recognize multi-person actions in the real world. Other investigators have addressed the problem from a case-based point of view [17].

Other researchers have followed somewhat different approaches. Charniak and Goldman [18] investigated Bayesian plan recognition. Tambe [19] and his colleagues [20,21] have focused on plan recognition of multi-agent systems involved in teamwork. Huber [22], Han and Veloso [23], Pynadath and Wellman [24}, Devaney and Ram [25] and Goldman, et al. [26], also

have contributed to advances in plan recognition. Our work presented here, however, focuses on the near-term intentions of a single agent — those that will manifest themselves within the next several seconds or minutes.

We now briefly discuss the work of Drewes, upon which our work is largely based. Drewes, et al. [27], in his prototype system TRAMS, presents what he calls Template-based Interpretation (TBI) for interpreting a human's intention in a simulation through external observation of this human's actions. His work involves observing the human's low-level actions and reflecting those actions within partially filled templates. These templates consist of attributes that represent low-level actions performed when that plan is being executed. As these low-level actions are performed by the human, the corresponding attributes are "checked-off." A template can include temporal and sequential relationships between different low-level actions. Each template reflects a plan potentially followed by the agent. As templates come closer to having all of their attributes checked off, they compete with each other for the right to be proclaimed as the one reflecting the agent's intent. His results on a prototype in the aviation domain indicate that this approach is able to correctly identify an agent's intent. However, one challenge not addressed by his work was the significant difficulties associated with creating the templates and their related weights. The investigation described here addresses exactly this issue.

Given that our work depends heavily on the concept of contexts and context-driven reasoning, it is worthwhile at this point to briefly mention related work in context-driven human behavior representation. Turner [28,29] used behavioral contexts arranged in hierarchies to control a robot's behavior. Both investigations used rules to recognize the environmental triggers to activate the behavior. Brezillon [30] and Bass [31] have also independently developed context-based approaches to modeling human behavior.

1.2 Specific Problem and General Approach

The review above indicates that whereas significant research is on-going in intention recognition, the problem has yet to be fully solved. We present an approach to recognizing in real time the near-term intention of an observed agent as early as possible in the execution of its actions. This determination should be done as quickly as possible to permit the observing party to predict the observed agent's future near-term actions as they unfold. This gives the observer maximum opportunity to counter the observed agent's actions. The system resulting from this work observes the agent (typically a vehicle controlled by a human) and, after noting the execution of one or several low-level actions, declares the intention of the agent. It does so by identifying its context — a module of knowledge that is able to control the agent in a particular situation. If the context in which the observed agent is operating is known by the observer, then it is relatively easy to predict the agent's future actions by modeling the agent with its active context.

Applications of this work exist in military tactical planning as well as operations, where inferring the intentions of an enemy is important. Furthermore, the application of a tank rounding a turn on the road also has more specific application in military affairs. For example, being able to accurately predict the position of an enemy vehicle can be helpful in targeting it. Nevertheless, the original motivation for this work was in live-virtual embedded simulations for training where live and virtual units find themselves on the same virtual battlefield. Knowing where a live vehicle will be at a specific time can reduce the required communications bandwidth in the live range. The last application requires significant accuracy in predictions.

The basis of our approach is *Template-based Interpretation* [27] and our extension of it, *Temporal Template-based Interpretation* [32]. This technique is described later in this article. To evaluate the feasibility of our approach, we use the prediction of the near-term driving pattern used by a tank driver as he rounds a turn on a road.

The following questions were addressed in our investigation:

- Can Temporal Template-based Interpretation be used to infer an agent's near-term intentions in real time by observing its actions and the situational parameters of the environment in which it operates?
- How can one efficiently build an artifact to perform this intention recognition? We specifically refer to the need to build the templates and assign weights to each and every low-level action attribute in the templates.

Our approach is founded upon recognizing actions associated with a previously defined template. We extend Drewes' work in several ways. The two most significant are 1) how we structure and manipulate the templates, making them capable of describing continuous actions and 2) how we arrive at the weights assigned to each attribute in the various templates. The first enhancement allows us to define a template attribute as a series of data points occurring sequentially over time. Thus, we call this enhanced version Temporal Template-based Interpretation (TTBI).

More importantly, whereas Drewes selected the values of the weights after consultation with experts and a significant amount of trial and error, we extract the weights from direct observation of prior agent behavior. Furthermore, we associate each template with

a context, as defined in CxBR. While contexts in CxBR are control mechanisms for an agent to successfully navigate a tactical situation, a template only contains the information about what could be externally observed about an agent being controlled by its corresponding context. This one-to-one correspondence between templates and contexts provides for integration of the two techniques and facilitates the prediction of the post-recognition behavior of the observed agent.

The objective of this research is to determine the technical feasibility of Temporal Template-based Interpretation for the general problem defined. The application used to evaluate the concept is only the test application and not the driving force behind the paper. A brief description of Temporal Template-based Interpretation (TTBI) is given in the next section.

2. Approach to Problem

Template-based Interpretation is a weak-model approach that makes real-time observations of an agent's performance and matches its current observed behavior to predefined templates of potential high-level intentions. The template that most closely accounts for the agent's observed actions (beyond a certain threshold level) is deemed the winner. We perform this match with a combination of single-layer, feed-forward neural networks with back-propagation training and fuzzy sets. We first briefly describe the original Template-based Interpretation concept, as it is an important component of our approach. Next, we (even more briefly) define the ideas behind Context-based Reasoning, as it is also influential in our work. We then discuss how, by combining the two approaches, we can accomplish our objectives. Then we describe our enhancement to TBI, called TTBI. Lastly, we explain how to automatically obtain the knowledge necessary to infer agent intention.

2.1 Template-based Interpretation

Template-based interpretation involves using models, or *templates*, of typical human behavior to infer the intention of a human or of an agent acting like a human. In some ways, it can be said to be an extension of case-based reasoning in that a template represents the pattern defining each case. The case/template most closely matching the pattern of the inputs is declared as the one most representative of the observed inputs. However, TBI extends traditional case-based reasoning by considering the temporal ordering of discrete events and the time differences between these events. Furthermore, TBI monitors the inputs continuously, looking for the execution of low-level actions by the agent being observed. Each template has selected attributes that represent actions that would be executed by the agent if it were carrying out the plan identified by that template, as well as aspects of the agent's state and of its environment. The attributes in a template include only those actions and aspects relevant to the intent represented by that particular template. At each monitoring cycle, each template's attributes are updated by an evaluation mechanism. When an action is observed, attributes that represent that action are "checked off" in each template that contains that particular action. This enhances the overall score for those templates containing that checked-off attribute in relation to the attribute's pre-assigned weight.

A template is not considered a candidate for identifying the agent's intentions/context until its overall score exceeds a minimum threshold value called the Critical Threshold (T_c). The first template to exceed the T_c is chosen as the one representing the observed agent's intentions/context. Figure 1 illustrates the components of the template-based interpretation approach. In this

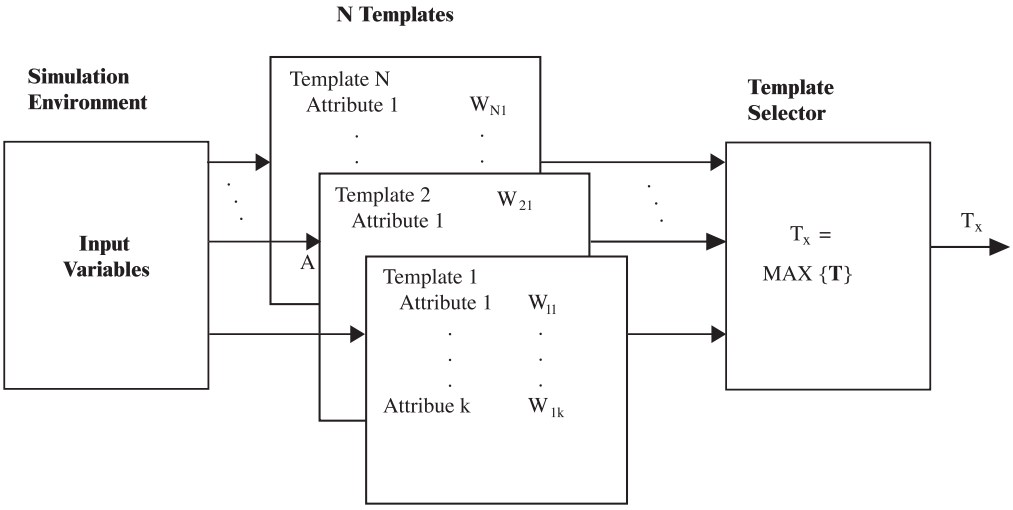

Figure 1. Template components

figure, W_{ik} is the weight associated with attribute k of template i, and T_x is the template from the set **T** of n competing templates that has the highest score above the minimum critical threshold value.

More formally, each template is a model for a specific high-level action. It describes the observable low-level actions that a *performing agent **A*** would do if it were indeed performing that high-level action. TBI works by defining the set of templates ***T***, so that

$$\mathbf{T} = \{T_1, T_2, T_3, T_4 \ldots T_n\}$$

where each template T_i contains several attributes. The attributes of one template are not the same as those of other templates, but they could have some attributes in common. Therefore, template T_i can be defined as consisting of several attributes atr_{il}, each describing an action made by agent **A** which partly indicates **A**'s current context, and which is observable by the observing agent **O**.

$$T_i = \{atr_{i1}, atr_{i2}, atr_{i3}, \ldots, atr_{ik}\}$$

Each attribute is assigned a Boolean value indicating whether the action described by it has been performed by agent **A**. If relevant, it also is possible to assign a time stamp for that action.

Furthermore, the importance of the attributes may not be equal for interpreting the actions of the agent **A**. Some may be more significant than others, and this can be indicated through a weight assigned to each attribute. Thus, attribute $atri_{il}$ for template T_i is a triple consisting of its weight, the Boolean "check off" and the time stamp.

$$atr_{il} = < \mathbf{W}_{il}, \text{YES/NO}, t>$$

where the weight, $\mathbf{W}_{il}$, is a real number between 0.0 and 1.0.

As agent **A** is being observed by agent **O**, the latter monitors the execution of low-level actions by **A**. Upon noticing that an action represented by atr_{il} has been executed by **A**, it will search each template in **T** and look for attribute atr_{il} in each template $T_i \in \mathbf{T}$. When it finds one, it will place a YES and a time stamp on the attribute. This can be likened to the game of Bingo, when a number is called and players place a game piece on the called number in every card where the called number is found.

As agent **O** continues to observe **A**, the numbers of "check-offs" on the various competing templates grow. Templates progressively become more completely "checked off" as more of the actions symbolized by their attribute are executed by **A**. Templates that truly reflect **A**'s intention will have more checked off attributes, which will translate to a higher overall score for these templates. This process represents a competition among the templates in **T** to be designated as the one truly describing the intention behind the actions of agent **A**. At some point, the cumulative weight of one template exceeds its critical threshold T_c, at which time, this template is considered the winner of the competition. The reader is referred to Drewes et al. [27], and Drewes [33] for details on TBI.

2.2 Context-based Representation of Behavior

The behavior of **A** can be said to be controlled by several behavioral contexts that prescribe its actions. These contexts, called C_i, are members of a set of contexts that define the behavioral universe for agent **A**.

$$C = \{C_1, C_2, C_3, C_4 \ldots C_n\}$$

Each such behavioral context contains the functions and procedures that result in agent **A**'s actions and decisions while in that context. One and only one behavioral context can be in control of the agent at any time. This context in control is called the *active context*, while all others are *inactive*. Each context also contains the knowledge of how to transition to other behavioral contexts that, over time, emerge as being more relevant to the situation currently faced by the agent. This is called *context transition*, and it is triggered by certain environmental events that signify that the situation faced by the agent has changed sufficiently to warrant activation of another behavioral context to better handle the emerging situation. This transition represents the simultaneous self-deactivation of the active context and the activation of another, more relevant context. *Context-based Reasoning* (CxBR) is a human behavior modeling technique that uses this approach to model human behavior in tactical situations. See [1] for more details on CxBR.

Therefore, if agent **A** can be said to be in context C_i, we represent this as $\mathbf{A}[C_i]$. Furthermore, if agent **O** can infer that agent **A** is operating under context C_i, and the description of C_i is known to **O**, then **O** can predict the future near-term actions of agent **A**. Our approach is based on doing exactly this — inferring context C_i in **A** by observing its actions externally. Therefore, while T_i and C_i are naturally associated, the viewpoints and functionality are radically different. This paper only addresses the first part of the process — inferring T_i.

2.3 Temporal Template-based Interpretation

The original version of TBI could only detect and record discrete low-level actions — e.g., putting down the landing gear, firing the main gun, etc. TBI is not capable of

interpreting intent if the actions indicative of such intent are continuous in nature and last for several minutes. An example of this would be that a drunk driver would be prone to continuously swerve her car on the road. Such actions cannot be identified or interpreted with a static attribute. TTBI permits the definition of temporal templates containing attributes defined as sequences of time- or distance-related data points. Incidentally, this would make the determination of the weights by hand very complex, providing the inspiration for the other major enhancement to TBI — the automatic assignment of weights based on observed behavior by an agent in a simulator.

To better correlate with corresponding behavioral contexts, TTBI implements a hierarchical organization of templates. The top-level templates are called competing templates. These templates can describe an agent's high-level intent, and correspond to Major Contexts in CxBR. Supporting templates, on the other hand, are used to provide attribute values if they are needed by a competing template or another support template. Competing templates are evaluated in a bottom-up fashion. Each supporting template is evaluated before the competing templates are evaluated. Template attributes can be *Mandatory* or *Non-mandatory*. A template is considered able to compete (referred to as being *valid*) if all of its Mandatory attributes (if any) are true, and unable to compete (*invalid*) if there is one or more Mandatory attributes, and at least one of them is not true.

Non-mandatory attributes contribute to the total value of the template. Their weights are indication of how relevant they are to the agent's perceived intent. However, they don't have to have a value or be true for their template to be considered valid. Valid competing templates whose output value exceeds T_c are compared immediately upon exceeding T_c, and the one with the highest value is announced as the current template representing the context active in agent **A**.

Mandatory attributes can be allowed to contribute to the value of the template if a non-zero weight is used. Conversely, if a zero weight is used for a Mandatory attribute, it contributes nothing to the output value of the template. It merely needs to be true for the template to be even considered in the competition. A zero weight for Mandatory attributes is typically more commonly used than non-zero.

2.4 Automated Weight Determination

Observing the structure of a template, one could make the analogy that each template is similar to a single layer neural network with a linear activation function. Specifically, the inputs (attributes) each are individually multiplied by their own weights and the summation of each of those results becomes the output. Figure 2 illustrates the analogy. Let I_k be the k^{th} input attribute and W_{ik} be the corresponding weight associated with the k^{th} input. Their product is summed with the other products for the output of the i^{th} template, T_i. This is equivalent to a single-layer neural network with a linear activation function and output T_i.

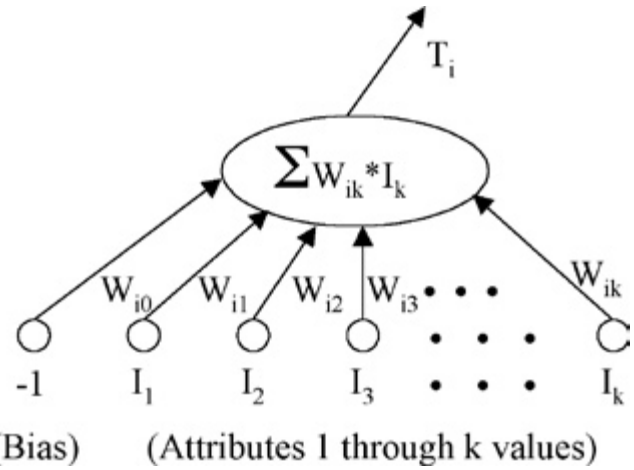

Figure 2. Template weight learning by observation using NN framework

That being so, a neural network training algorithm, such as back-propagation, could be used to set the value of the weights [34]. For a one-layer neural network, this algorithm is called LMS. In this case, the template output value, the sum of the weights of each input attribute multiplied by its associated weight, corresponds to the output of the neural network with a linear activation function.

With this intuition, we used the neural network training procedures for determining these template weights by presentation of examples of the complex behavior to be recognized. Additionally, the templates were not restricted to a single layer neural network for implementation. For each template, only the template output resulting from the inputs is a consideration. This is because the important thing is that the output from each of the competing templates has to compare correctly against each of the other templates. Specifically, the template that wins the competition should be the one with the highest output value, or score. Furthermore, multiple internal neural network layers would allow for greater output discrimination with more complex behaviors than could be possible with a simple summation of weighted values.

2.5 Fuzzy Functions for Attributes

For training neural networks to recognize the complex behavioral contexts, a means for representing and

normalizing the attribute input values was needed because the neural networks train most efficiently when the input values are in the range of -1.0 to +1.0. The approach taken was to use fuzzy sets to represent the membership of the attribute to the behavior pattern classification whose detection is being sought. As Zadeh [35] points out, fuzzy sets provide a natural way for handling problems where sharply defined criteria for class membership are absent. That is the situation here, since specific attributes could be members of more than one template. In other words, we use fuzzy membership functions to define the weights.

Fuzzy sets are characterized by a membership function that assigns a grade of membership between 0.0 and 1.0 for each member of the set. A membership value of 1.0 indicates full membership, while 0.0 indicates no membership. Values in between indicate partial membership. Langari and Yen [36] provide some examples of membership functions based on the exponential function. One of their membership functions is shown below for the membership of the parameter T to the class "medium."

$$\mu_{medium}(T) = exp\ (\ -\ \alpha\ |T - T_0|^{\ p}\ ),\ p \geq 1$$

The above equation provides a description for a class where the membership, $\mu_{medium}(T)$, is highest for the parameter T around a given value T_0, and decreases as the parameter deviates from that value. T_0 denotes the center value where the membership is strongest. The scaling factor α affects how broadly or narrowly the membership function is defined.

The values of the inputs for each set of behavior patterns to be recognized were statistically analyzed to determine the mean value and standard deviation for each parameter for the examples representing each pattern type. The mean corresponded to the T_0 in the above equation, and the inverse of the standard deviation was used as a scaling factor to define how narrow the membership function would be. Thus, each input parameter for neural network training was transformed to a range of 0.0 to 1.0 through a fuzzy membership function. We now describe an application of this approach that identifies the intention of the observed entity and can determine the value of the weights through observation.

2.6 High-level Algorithm Describing this Approach

The approach presented in this paper can be described by the following high-level algorithm:

1) Observe the entity to be modeled either in a simulator or in the real world. Record the values of all variables of interest. There will be one data set for each run executed.
2) Classify the results of the observation into types of actions that were executed by the performing agent **A** during the observation phase.
3) Design a membership function for each of the identified action types. Transform each data set with each of the membership functions defined. Assign a value of 1.0 or 0.0 to each data set depending on whether or not the transformed data set describes the intention/behavior corresponding to each membership function.
4) Partition the data sets into training, validation, and testing data in accordance with established procedures.
5) Design the architecture of the neural networks to be used according to established procedures.
6) Train neural networks to determine the weights of each template attribute.

We should note that no effort was made in this investigation to automate the selection of the variables of interest. In our evaluated application, the variables of interest were selected by a human who considered what was strictly necessary to infer the agent's intention. However, we see variable selection as an important issue, not only to select those variables that are necessary, but also to leave out those that are not. We leave this for future research.

3. Evaluation of Approach

To evaluate the effectiveness of our approach, we implemented TTBI in an application for predicting how an agent is driving a vehicle in a military simulation. The vehicle in question is a battle tank (M1A2) and the task is how to navigate a turn with this vehicle in a simulated environment. There are many ways experienced tank drivers can perform this maneuver. This application calls for an accurate prediction of a simulated tank's path around a turn. The accuracy requirements are indeed unforgiving. Lateral deviations (positive to the right; negative to the left) are limited to no more than 14.4 inches. Motional discrepancies (forward being positive) were limited to less than 29 inches. Our use for such accurate prediction was to reduce the communication bandwidth required in a distributed simulation by being able to predict accurately the location of the vehicle [37].

To eliminate the difficulties involved with sensors interpretation in the real world, we restrict the performance of agent **A**'s behavior to a simulation. Furthermore, we restrict our evaluation to recognizing the agent's near-term intentions to follow a particular path when driving the tank around a turn in the road. The selected path is one of several paths pre-classified through observation of the agent's past behavior. While

this admittedly does not involve recognition of high level-intentions *per se*, it does provide the ability to predict the exact location of the agent by recognizing the path upon which it is embarking at an early stage of the action. It represents a special case of the general case described in Section 2, but with little loss of significance. In effect, the template has several sub-context templates vying for selection as the one the agent is executing. Given the structural and functional similarity of major context templates and sub-context templates, the evaluation of how TTBI infers sub-context level templates does not in any way invalidate the evaluation of the TTBI method as it would approach the major context template.

3.1 Simulation Infrastructure Used

The simulation system used for this experiment was the ModSAF system, a constructive military simulation environment. A specific turn was selected in the terrain database for the National Training Center in California. Figure 3 depicts the turn, all its related parameter definitions, and a typical outside path taken by the simulated tank. ModSAF contains the functionality to steer a virtual tank though a turn, albeit imperfectly.

The road segment of interest here is defined by its *waypoints*. Waypoints typically are placed in locations that mark a change in direction for the vehicle of interest. A *turn*, therefore, is defined as a segment of road consisting of three waypoints that are not in line with each other. Figure 3 depicts three of them, one at each terminus and one at the center of the turn. Depending on the angle of the turn, the driver may choose to take the turn either on the inside, thus cutting off the angle, or on the outside, thereby curving wide around the turn to provide room for maneuver. How the tank merges back into the center of the road after the turn also can vary in many ways. Therefore, the tank's path is decomposed into two phases — its *approach* to the waypoint, defining the turn of interest, and its *departure* from that waypoint toward the next waypoint. The approach is the path taken by the agent prior to reaching the center waypoint. The departure is its path thereafter. The objective is to predict where the simulated tank (agent ***A***) would be at all times within the strict accuracy requirements stated above. We accomplished this by observing a priori that there is only a handful of ways the turns are taken, and later classified the real time data into one of these paths using TTBI.

3.2 A Priori Observations of Agent A in the Simulation Environment

To learn the weights to be used in the templates, 110 simulations runs of the agent A taking the designated turn were executed in ModSAF and recorded. Only these 110 recorded paths were used to learn the values of the weight of the template attributes. The headings of agent A at the start of these simulations varied by one degree, from 6° to the left of north to 4° to the right of north. Ten runs were executed for each initial heading, with each run beginning 20 waypoints and 3 km before the designated turn. Given the slightly different initial headings, the actual paths taken by agent A tank in ModSAF varied significantly, but all within the realm of realism as if executed by a human.

Analysis of the results indicate that the approaches can be classified into three categories: 1) Early — the tank began to curve away from the road centerline more than 45m before the turn center waypoint; 2) Nominal — between 30m and 45m before the turn waypoint; and 3) Late — less than 30m. Furthermore, each category could be further divided into single curve and double curve. Table 1 shows the distribution of the classifications, indicating that early and nominal single curves were the most prevalent. Figure 4 shows

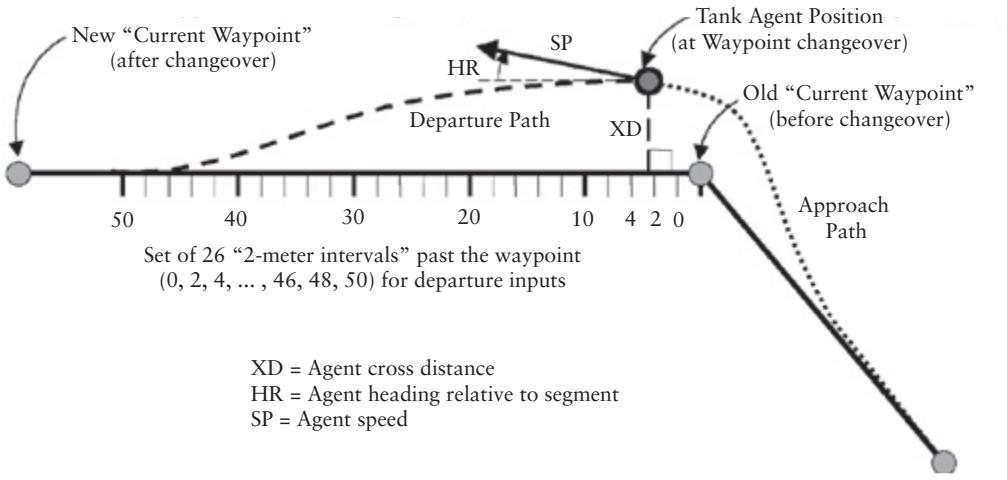

Figure 3. Turn used for experiments

Approach Categories	No. of Runs
Early, Single Curve	47
Early, Double Curve	15
Nominal, Single Curve	41
Nominal, Double Curve	0
Late, Single Curve	7
Late, Double Curve	0
Total	**110**

Table 1. Summary of approach category frequencies

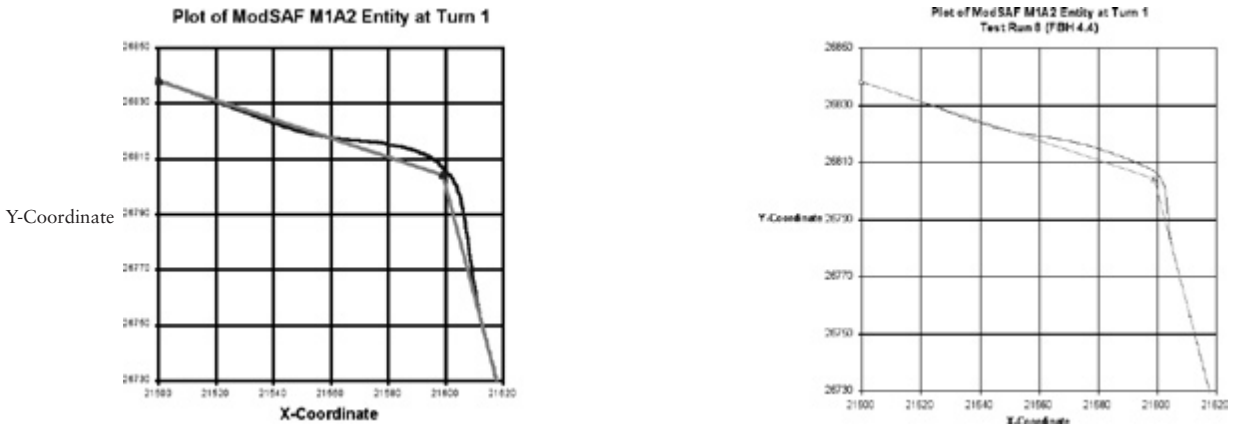

Figure 4. X-Y coordinates of sample experiemental runs

Departure Types	No. of Runs	Description
Flat Nominal (FN)	15	Parallels outbound segment, then turns to intercept at about 45 meters beyond the turn
Flat Bow Low (FBL)	16	Same as FN, but slight bow before turn to intercept
Flat Bow High (FBH)	17	Same as FBL, but slightly more pronounced bow
Bow Nominal (BN)	35	Definite bow on the departure that continues to an intercept at about 45 meters beyond the turn
Straight Angled (SA)	10	After initial turn, maintains an intercept heading with little or no change to an inctercept beyond 80 meters
Bow Wide (BW)	7	Much wider bow than BN with intercept beyond 80 meters
Bow Asymptotic (BA)	3	Initially similar to BN, then smoothly reduces intercept heading to asymptotically intercept beyond 60 meters
Bow Distant (BD)	3	After initial turn, slowly turns to an intercept heading beyond 80 meters resulting in an extended slight bow
Double Curve (DC)	3	Initially turns to a heading that would intercept close to the turn, then turns away from the segment followed by a turn back that closely resembles the SA pattern
Close Intercept (CI)	1	Intercepts within about 10 meters followed by a long-lasting overshoot before eventual re-intercept

Table 2. Departure pattern types

two data samples — graph A depicts an early, single curve approach while B shows a nominal single curve approach.

The departure categories were somewhat more complex. Ten different categories were identified. Table 2 contains a summary description. See Table 11 in [32] for details. Note that these pre-classifications were done manually by the investigators. However, the work could be reasonably extended in the future to do this automatically with a clustering algorithm.

3.3 Extracting the Weights from the Observations

Training, validation, and testing examples were created from the 110 simulation runs. These runs were used to train neural networks to recognize the six most frequently occurring departure types on the turn. First, the runs for the departures were selected and the XD, HR, and SP variables (defined in Figure 3) were calculated for each run. Then, six fuzzy set membership functions were created from those variables by calculating the means and standard deviations for the XD, HR, and SP variables at 26 locations for the each of the six different data sets. These 26 locations were at the beginning of each 2-meter segment from the turn's center waypoint (its knee) to the third, as shown in Figure 3. The runs in each data set included only those runs of a specific departure type, e.g., 35 BN categorized runs were used for the BN set and 17 FBH runs were used for the FBH set.

The fuzzy membership functions expressed how close the values for the individual runs were to the average value for that variable for that classification data set. These membership functions were based on the Gaussian probability density function, given by the equation below:

$$f(y) = \left(1/\sigma\sqrt{2\pi}\right) e^{-(y-\mu)^2/2\sigma^2}$$

where μ is the sample mean and σ is the sample standard deviation of a normal random variable y.

The constant in front of the exponential term was removed because for the transformation function, the desired output is 1 when the variable is equal to the mean (y = μ), and approaches 0 when |y - μ| is not close to the mean (i.e., the value is much greater than σ). In the resulting Gaussian fuzzy function shown below, the variable Z = (y - μ) / σ is a measure of the number of standard deviations that the variable is above or below the mean value.

$$ff_G(y) = e^{-Z^2/2}$$

After creating those six fuzzy functions, they were applied to the selected data for each of the 110 runs, thus creating six master data sets of transformed inputs for each of the 110 runs, regardless of which type they were. Within each of the six master sets, those runs that were classified as being of the same type as the fuzzy function used for the transformation had their single output value set to 1.0. All other runs in the set had their single output value set to 0.0. The runs from those data sets were then separated into training, validation, and testing sets. The neural networks were then trained with the training and validation sets for each of the six classes, and their learned weights were the attribute weights for the template they represented.

The format for the data set examples for the three data values (XD, HR, and SP) for the 26 locations (from 0 through 50 meters past the center waypoint) is a sequential representation of the three variables in each of 26 rows with the output value (1.0 or 0.0) as the single data item on the 27th line. Table 3 shows examples for a run with an FBL departure type at the turn. The raw, calculated values for XD, HR, and SP are in the box on the left with the nominal values of DA, the distance after the turn, next to the applicable rows. The two examples on the right were transformed using the FBL and BN fuzzy membership functions, respectively. Note that there are 26 examples produced from each run, one for each 2-meter increment of DA, since the XD, HR, and SP values are zero at each DA value until the agent has reached it. The 1.000 at the end of the middle box indicates that the data indeed represents an FBL departure. On the other hand, the 0.000 at the end of the right box indicates that the raw data in the left box do not represent a BN departure.

3.4 Templates for Application

The context and sub-context templates developed for this research addressed only the limited case of a *Road March* task — the tactical process followed by the military to move men and materiel from one place to another by way of a road, without enemy presence. As such, there was only one major context template created — **RoadMarch**. Nevertheless, separate template files were created for both the **RoadMarch** major context and the sub-contexts relevant to the **RoadMarch** major context. The experiment, however, centered on identifying the sub-context controlling the tank entity. Given the identical structure of sub-contexts and major contexts, this was not considered a limitation to the experiment.

The **RoadMarch** major context template is simplified — it contains only one attribute and it is mandatory. That attribute, *NearRoad*, is true and produces a template value of 1.0 if the current value of XD, the

	Raw Values			FBL Gaussian Transform			BN Gaussian Transform		
DA	XD	HR	SP						
0	2.283	7.348	5.138	0.906	0.865	0.949	0.999	0.232	0.077
2	2.467	7.348	5.518	0.966	0.865	0.908	0.473	0.232	0.218
4	2.730	5.167	5.879	0.999	0.991	0.879	0.313	0.054	0.367
6	2.909	5.167	5.922	1.000	0.696	0.950	0.213	0.010	0.272
8	3.039	3.228	6.258	0.998	0.845	0.907	0.145	0.304	0.389
10	3.134	1.549	6.354	0.998	0.872	0.964	0.112	0.241	0.333
12	3.193	1.549	6.598	0.997	0.842	0.951	0.101	0.999	0.375
14	3.231	0.144	6.883	0.995	1.000	0.883	0.094	0.993	0.429
16	3.236	0.144	6.935	0.995	0.955	0.984	0.111	0.550	0.320
18	3.222	-0.972	7.220	0.994	0.950	0.953	0.150	0.675	0.449
20	3.189	-0.972	7.264	0.997	0.950	0.980	0.232	0.121	0.381
22	3.155	-0.972	7.550	0.997	0.982	0.950	0.323	0.030	0.616
24	3.120	-0.972	7.930	0.996	0.812	0.957	0.424	0.139	0.898
26	2.991	-5.721	8.311	0.999	0.945	0.996	0.481	0.844	0.995
28	2.821	-5.721	8.596	0.997	0.945	0.980	0.578	0.360	1.000
30	2.553	-9.310	8.629	0.996	0.875	0.988	0.684	0.882	0.976
32	2.272	-9.310	8.629	0.999	0.915	0.874	0.770	0.733	0.934
34	1.903	-10.763	8.189	0.982	0.807	0.990	0.817	0.866	0.625
36	1.493	-10.763	8.189	0.946	0.830	0.873	0.830	0.874	0.678
38	1.186	-10.763	8.189	0.978	0.583	0.821	0.887	0.878	0.679
40	0.823	-9.410	7.990	0.936	0.809	0.910	0.909	0.789	0.636
42	0.473	-9.410	7.990	0.876	0.792	0.887	0.893	0.999	0.762
44	0.182	-9.410	5.846	0.855	0.788	0.135	0.892	0.998	0.001
46	-0.187	-11.930	3.259	0.743	0.644	0.325	0.833	0.725	0.000
48	-0.416	-0.715	2.193	0.738	0.438	0.278	0.851	0.045	0.000
50	-0.437	-0.491	2.903	0.918	0.547	0.255	0.984	0.170	0.338
				1.000			0.000		

Table 3. Gaussian transformed examples

entity's cross distance measured perpendicularly from the current road segment's centerline, is less than 5.0 meters on either side. This equates to a simple rule: "If the entity is within five meters of the road segment's centerline, then the **RoadMarch** template is valid with a competing value of 1.0; else, it is invalid and does not compete." This is reasonable. Otherwise the tank agent would not be following a road and would be involved in some other action.

The sub-context templates that competed for selection were divided into two groups — approach templates and departure templates. The approach templates (A_EAR, A_NOM, A_LATE, and A_DBL) identified the type of approach the agent being observed (**A**) was taking. These approach templates are rule-based, and the rules turned out to be quite simple: If the agent began to separate from the centerline of the road more than 45m before the center waypoint, then it was classified as early (A-EAR). If the separation began between 45 and 30m before the center waypoint, then it was considered nominal (A_NOM). If separation began closer than 30m, it was late (A_LATE). The double curve — a double "hump" as it separated from the road centerline — introduced a bit of intrigue into the process, but in general, these templates were too simple to be interesting and were not pursued any further.

The departure templates, on the other hand, did present interesting challenges. These used neural networks with weights trained by observation of examples of the behaviors as described earlier. They are described in this next section

3.4.1 Neural Network-based Competing Templates for Turn Departure Sub-contexts

There were six competing departure templates based on the observed behavior categorized into sub-contexts for the **RoadMarch**. The six departure templates were **DepartureOutsideBN** (DP_BN), **DepartureOutsideFBH** (DP_FBH), **DepartureOutsideFBL** (DP_FBL), **DepartureOutsideFN** (DP_FN), **DepartureOutsideSA** (DP_SA), and **DepartureOutsideBW** (DP_BW). The "DepartureOutside" label indicates that in each of these categories the agent took a wide turn on the outside road,

as opposed to cutting off the turn inside to minimize the distance.

The only mandatory attribute, which calls the support template DPOUT to return its validity, is the same for all six departure templates. DPOUT indicates whether the departure is outside or inside. Its weight is set as 0.0, so this mandatory attribute contributes nothing to the confidence value of the template, but is necessary for the template to be considered for competition. The DPOUT support template can be valid only as long as the entity is closer to the previous waypoint than the current one being approached. This identifies that the entity is on the departure phase of the turn, and not its approach. Thus, none of these competing departure templates can be valid at the same time as the competing approach templates.

Two non-mandatory attributes respectively call two functions that return the *History* and the *Recent* neural network outputs for the departure type of the template. For example, the DP_BN template calls the BN_NN_HISTORY and BN_NN_RECENT function attributes and the DPFBL template calls the FBL_NN_HISTORY and FBL_NN_RECENT function attributes. The History neural network takes advantage of all the data being built up as the entity progresses through the turn. In other words, the more ground the observed agent has covered, the more certain the observer becomes that it is properly identifying the turn classification. However, because of the nature of neural network training with examples that are more heavily filled in with the earliest data in the departure, a change in behavior part way through the departure would not easily overcome the earlier identification. The History neural networks were trained to output 1.0 for its recognized turn type and 0.0 for all other types.

The Recent neural network, on the other hand, focuses on a moving window of the most recent data points in order to counteract the inertia of the History neural network. They were trained to output 1.0 for recognized turn types and -1.0 for all others. The attribute to which the History neural network is assigned has a Certainty Factor (CF) value of CFH, while the attribute assigned the Recent neural net has a CF of CFR.

Those returned attribute values, when multiplied by their individual weights (CFH and CFR, respectively), are combined using standard certainty factor procedures to produce a template confidence value. That value is then compared to the template's critical threshold, T_c, to determine whether the template can be considered the winner if it has the highest total output. The values to use for the weights and T_c were determined by experiment, which will be discussed in the next section on testing.

4. Testing and Evaluation of Prototype

Performance of the prototype was assessed using two measures: 1) number of correct identifications of the path actually taken by the observed agent and 2) how early a correct identification of the path could be made. There were two series of tests performed on the prototype system developed as part of the work described. In part 1, the template evaluation mechanism was subjected to the previously collected data representing an agent making a series of runs not used in training the neural networks, but for the same turn on which the training data was obtained. This we refer to as *Turn #1* testing. The second set of tests evaluated the generalization ability of the system. We subjected the same templates to runs executed on a similar, but different turn in the same National Training Center terrain database. We refer to this as *Turn #2* testing. Runs from Turn #2 were not used in training the neural nets.

Of the 110 (non-repeatable) runs used as data for our work, 18 of these, arbitrarily selected, were designated for use only during testing, and not used for training or validating the neural network. Each run was composed of the approach phase and the departure phase. As mentioned above, however, the approach phase was deemed almost trivial and thus was not evaluated. More interesting was the departure, as there were several different types and they were not easily distinguished from one another by simple rules. This evaluation of the departure portion of the turn formed the basis for our testing. We begin by describing Turn #1 Testing.

4.1 Turn #1 Testing

The 18 runs saved for system testing were evaluated using the templates and their accompanying neural networks for identification. The output of the competing templates (six in all) was a value between -1.0 and 1.0. The value indicated the confidence that each template had that the current run being presented to them was of their type. The results were evaluated as either *Correct ID*, *False ID* or *No ID*. A Correct ID was indicated when the winning template in the template competition agreed with the a priori classification of the test run. A False ID result indicated disagreement. A No ID came about when the template competition did not offer any template that arose above the T_c value selected for that competition. One must note that in two of the 18 test cases, the correct classification of the run was not found among the six template classifications. This happened because there were more than six original classifications, but only the six most popular in terms of frequency of

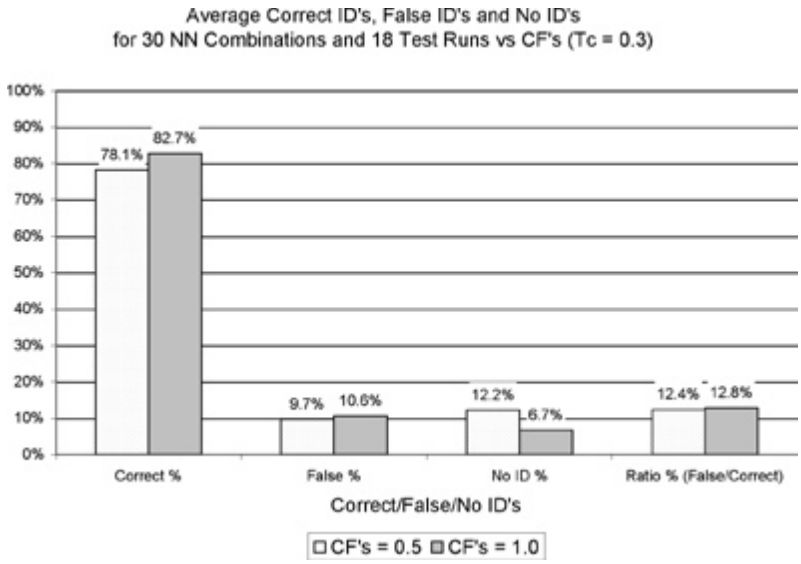

Figure 5. Averages of correctness results versus equally weighted CFH/CFR

DA	BN	FBH	FBL	FN	SA	BW
0.0	-1.00	-1.00	-1.00	-1.00	-0.97	-0.93
2.1	***0.87***	-0.99	-1.00	-1.00	-0.97	-0.95
4.3	-1.00	0.01	-1.00	-1.00	-0.97	-0.99
6.2	***0.93***	-0.73	-0.98	-1.00	-0.97	-0.99
8.1	***0.91***	-0.74	-0.99	-1.00	-0.97	-0.99
10.2	***0.93***	-1.00	-1.00	-1.00	-0.97	-0.99
12.2	***0.99***	-1.00	-1.00	-1.00	-0.97	-0.99
14.3	***0.99***	-0.99	-1.00	-1.00	-0.97	-0.99
16.5	***0.99***	-1.00	-1.00	-1.00	-0.97	-0.99
18.1	***0.99***	-1.00	-1.00	-1.00	-0.97	-0.99
20.3	***0.99***	-1.00	-1.00	-1.00	-0.97	-0.99
22.4	***0.99***	-1.00	-1.00	-1.00	-0.97	-0.99
24.1	***0.99***	-1.00	-1.00	-1.00	-0.97	-0.99
26.3	***0.98***	-0.97	-1.00	-1.00	-0.97	-0.99
28.5	***0.98***	-0.99	-1.00	-1.00	-0.97	-0.99
30.2	***0.95***	**0.75**	-1.00	-1.00	-0.97	-0.99
32.4	-1.00	***0.82***	-1.00	-1.00	-0.97	-0.99
34.1	-1.00	-0.99	-1.00	-1.00	-0.97	-0.99
36.4	-1.00	-1.00	-0.72	-1.00	-0.97	-0.99
38.1	***0.66***	-0.99	0.14	-1.00	-0.97	-0.99
40.4	***0.98***	-0.99	-1.00	-1.00	-0.97	-0.99
42.1	-0.12	-0.93	-1.00	-1.00	-0.97	-0.99
44.4	**0.70**	***0.92***	-1.00	-1.00	-0.97	-0.99
46.1	***0.96***	-0.69	-1.00	-1.00	-0.97	-0.99
48.3	***0.97***	**0.92**	-1.00	-1.00	-0.97	-0.99

Table 4. Competing template values (Turn1, BN Test Run r4.2). Note: Competitive templates are bolded and winning templates are italicized as well.

appearance were formalized and used in the evaluation. Therefore, for those cases not represented by a template, the correct identification in fact should have been No ID. For those two cases, if the template competition returned a No ID, this was evaluated as a Correct ID.

Figure 5 depicts the results obtained with CFH and CFR, certainty factors for "History" and "Recent" neural network outputs (which acted as template attributes), both being set to either 0.5 or 1.0 each, and T_c set at 0.3. This mix of CFH and CFR maximized the ratios of correct percent over false percent compared to other mixtures of CFH and CFR with values of 0.0, 0.5, and 1.0. These results indicated that when CFH = CFR = 1.0 rather than CFH = CFR = 0.5, the Correct IDs increased a bit, as did the False IDs, both at the expense of the No IDs. This difference is not seen as significant.

Additional experiments were performed varying the value of T_c. The objective here was to determine how selective to be in the template competition. Naturally, we expected the number of No IDs to increase as T_c was increased. This is in fact what happened. The Correct IDs and False IDs, however, remained relatively constant until T_c reached 0.70.

With regard to how early the identifications were made, we noted the first instance when the correct identification was made for those runs for which a Correct ID was made. Table 4 indicates an example of how early it was in one case to identify correctly the path taken by the simulated test entity. The results indicate that for the most part, the correct template was identified very early in the process, as in the first 2 to 6 meters beyond the turn waypoint. This is a tremendous advantage in that it

permits early identification of observed behavior, giving the observer time to react to the intended actions. This is particularly important in conflicts where knowledge of an opponent's intention can lead to better counter tactics and/or preparation.

The results for a single 78-5-1 History NN and a T_c of 0.3 on the 18 runs are depicted in Table 5. (78-5-1 indicates 78 input nodes, 5 hidden nodes, and 1 output node.) The early recognition is admittedly a characteristic of the data presented to the system, and it may not be the case in all applications. Nevertheless, when the data did permit early identification, the system was capable of doing so.

DA	Correct ID	False ID	No ID
0-2	72.2%	22.2%	5.6%
2-4	77.8%	16.7%	5.5%
4-6	72.2%	16.7%	11.1%
6-8	77.8%	11.1%	11.1%
8-50	83.3%	11.1%	5.6%

Table 5. Real-time recognition of agent intention for Turn #1

4.2 Turn #2 Testing

These tests were conducted with runs from the agent taking a similar, but not identical, turn in the same database. This test introduced several differences in how the test was performed. First of all, since we used the networks trained on data obtained for Turn #1, the idea of using training, validation, and testing runs was not applicable. Because the 110 original runs were executed for the entire route (encompassing both Turns #1 and #2 as well as other terrain), the same 18 runs were used for testing on Turn #2, except that the data specific for Turn #2 were used. Secondly, since the departure classes on Turn #2 were not categorized as they were for Turn #1, there was no automatic means to check for Correct ID, False ID, or No ID for the outputs. Thus, the winning template outputs from combining the History and Recent NN attributes could not be evaluated directly for correctness, as was done for Turn #1. Instead, the template program used for testing on Turn #1 was modified to record the winning template number along with the details of the template outputs rather than information on the correctness of the output. For Turn #2 test evaluation, the XD, HR, and SP outputs for each test run were compared against the XD, HR, and SP mean values for the departure type identified as the winning template output for each of the 18 runs to check for reasonableness.

Table 6 depicts the 18 runs on Turn #2. It shows that most of the runs could be labelled FBL, at least for the first half of the road segment after the waypoint. Two others could be categorized as FBH and one (run #2) had no similarity whatsoever to the classifications determined in Turn #1.

These determinations of what the templates should reasonably show were used as the basis for comparison to what the templates actually produced. In making this evaluation, ten sets of trained History neural networks and ten sets of Recent neural networks were used in various combinations so that each set was used three times with three different sets of the other type of neural network. Each set of trained neural networks consisted of six neural networks, each trained for a different one of the six major categorized turn types. Thus, 30 evaluations were made of each of the 18 repeatable test runs. The percentage results of correct, false, and No IDs for these 30 evaluations for each test run are shown in Table 7. Overall, approximately 84.9% of the template responses were correct, including No IDs when no match should have been made; 11.1% were false; and 4.1% were of No ID, when a match should have been made.

While these results do not provide the fidelity for the correctness average shown for Turn #1, it gives us an appreciation for the ability of the system to generalize, to some degree, and extend the results of one turn to another, similar one. The results indicate that, except for run #2, the system was able to identify the path taken by the simulated entity for at least the first half of the trajectory past the initial waypoint. Thereafter, the path diverges from any known categorization of Turn #1, and no identification was possible.

Continuing with the evaluation of how early the system can identify the agent's intent for Turn #2, Table 8 depicts the results tabulated for only the first 10 meters of Turn #2 departure. We used 30 sets of History and Recent NN's combined using CFH and CFR = 0.5 and T_c = 0.3 on the 18 test scenarios used for Turn #1.

4.3 Summary of Testing

The evaluation of the intention recognition of agent A on Turn #1 was generally good overall. Correct identifications were in the 75% to 85% range, false identifications in the 3% to 13% range, and no identifications in the 7% to 12% range. On Turn #2, the ModSAF agent used to generate the data did not reproduce the same behaviors that were identified on Turn #1. However, the early portions of most of the test runs were very similar to the early portions of one of the behaviors identified on Turn #1. If the identifications for the portions of the departures that were judged to successfully match are counted as correct and those

that did not as false, the off-line responses on Turn #2 would be 85% correct, 11% false, and 4% with no identification. These results are generally within the ranges noted for Turn #1.

None of the turn action types observed on Turn #2 were the same as the ones observed on Turn #1. However, the portions of the turn actions on Turn #2 that were very similar to an observed action type on Turn #1 were identified as that similar behavior. That showed reasonable generalization by recognizing portions of actions that matched what had been previously learned on Turn #1. There was a small but finite group of identifications of action types for the portions of the Turn #2 actions that did not match the learned actions from Turn #1. It is also seen that TTBI avoids incorrect identification of behaviors (including No IDs) where significant ambiguities existed and thus classification was not appropriate.

Test Runs	Departure Types	Location (Meters)	Comments
1	FBL	0 – 24	
4, 6-13, 15	FBL	2 – 24	
16, 17, 18	FBL	0 – 28	
3	FBL	8 – 28	
5	FBL	0 – 6	Only departure type in this region
	FBH/FBL	8 – 16	FBH is best match, but either is acceptable
	FBH	18 – 24	Only departure type in this region
2	None	0 – 50	No departure type matches expected anyplace
14	FBL	16 – 24	

Table 6. Summary of test runs at Turn #2 evaluations. Note: Lack of a winning template outside the locations shown is counted as a Correct ID

Test Runs	Correct ID (%)	False ID (%)	No ID (%)
1	86.9	12.3	0.8
2	42.9	57.1	0.0
3	91.5	3.8	4.6
4	88.6	10.6	0.8
5	76.4	9.7	13.8
6	82.7	10.8	6.5
7	92.6	1.3	6.2
8	94.2	3.5	2.3
9	92.6	1.9	5.5
10	95.4	3.8	0.8
11	91.9	6.5	1.5
12	93.8	0.8	5.4
13	91.5	1.9	6.5
14	66.0	25.0	9.0
15	93.3	1.3	5.4
16	83.1	16.9	0.0
17	83.5	16.5	0.0
18	80.4	15.0	4.6
Average	**84.86%**	**11.05%**	**4.10%**

Table 7. Correct, false, and no ID percentages for test runs at Turn #2

DA	Correct ID	False ID	No ID
0-2	76.7%	16.7%	16.7%
2-4	72.8%	7.2%	20%
4-6	93.3%	5.0%	1.7%
6-8	94.4%	5.6%	0.0%
8-10	87.2%	7.8%	5.0%
10-12	89.6%	8.7%	1.7%

Table 8. Real-time evaluation for Turn #2

5. Summary and Conclusions

The results of our evaluation of our weak-model approach to recognize the intentions of a tank driver to follow a certain path around a turn in the road proved to be largely successful at identifying its near-term intentions and predict its lower level actions. However, we believe that the similarity in structures to higher-level identification makes it such that the TTBI approach could be similarly applied at any level of context abstraction.

The intuition of equating the template attribute weights to neural network weights resulted in a good method to learn the weights directly from observation of prior agent behavior. This is critical for a weak-model approach such as presented here. While the process is not completely automated, future research could more readily make this learning process highly automated by using clustering algorithms to group similar types of runs.

In conclusion, while the method yielded good results, it required significant manual effort to review the runs and classify them, build the fuzzy set membership functions and train the neural networks. Much of the effort was application-specific. Table 9 lists important characteristics for generalizing the TTBI technique to other applications. These characteristics impact the feasibility of TTBI, as well as its context identification process during operation. Each characteristic is determined either by the knowledge engineer at design time, or at runtime by the constraints of the operational environment and entity operation itself. As listed in Table 9, the knowledge engineer must be able to decompose the behaviors into mutually exclusive contexts that are complete and consistent. It is important for the knowledge engineer to select the most indicative environmental variables for observation and a suitable level of behavioral abstraction. Automating the selection of these variables is beyond the scope of this research and is the subject of future research. Furthermore, the operational environment needs to be properly instrumented to gather the observation stream required by TTBI and it must do so in real time. Finally, the entity's behavior needs to exhibit all modes during training while exhibiting minimal unforeseen or inconsistent modes during operation. Making the system noise tolerant also is beyond the scope of this investigation and the subject of future research.

Lastly, we present a discussion about the computational complexity for scaling this concept to other major behavioral contexts and their behavioral sub-contexts for the extended TTBI approach. Let N denote the number of major context templates, with an average number of attributes being A. The number of comparisons to determine the major context would be on the order of $N*A$, or $O(N*A)$. If the number of attributes per template were considered relatively constant, the complexity would be expressed as $O(N)$. For any given major context that was selected as the winning template, the sub-contexts that would compete are limited to the sub-contexts of that major context. Thus, if the number of sub-context templates and their attributes were relatively constant from one major context to another, the complexity still would be expressed as $O(N)$. Therefore, the addition of new major contexts to be considered would scale with a linear complexity thereby demonstrating good potential for scalability.

Important Characteristics	**Responsible Information Source**	**Criticality for Success**
Ability to decompose entity behaviors into mutually-exclusive contexts within the given domain	Knowledge Engineer	Essential: must be accurate, complete, and internally consistent
Ability to identify which of all possible observable environmental variables should be monitored as being most indicative of current context/transitions	Knowledge Engineer	Important: very strong positive correlation desired, but set of variables selected need not be optimal for TTBI to work correctly
Ability to select a tractable yet useful level of abstraction of the entity's actions	Knowledge Engineer	Important: low-level behaviors can be more readily correlated with environmental observations
Availability of instrumentation to observe the identified environmental variables to create a stream of discrete or continuously-valued observations	Operational Environment	Essential: required during both the training and performance phases
Sufficient breadth and consistency in the entity's behavior as quantified by template inputs	Entity Operation	Important: lack of breadth during training or inconsistencies during operation will decrease the context identification rate

Table 9. Factors affecting the applicability of TTBI technique

6. References

[1] Gonzalez, A. J. & Ahlers, R. 1998. Context-based representation of intelligent behavior in training simulations. Transactions of the Society for Computer Simulation, 15(4), 153-166.

[2] Schmidt, CF, Sridharan, N. S. and Goodson, J. L. 1978. "The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence", Artificial Intelligence, Vol. 11, No. 1&2, pp. 45-83.

[3] Clark, A. N. 1994. Pattern recognition of noisy sequences of behavioral events using functional combinators. The Computer Journal, 37(5), 385-398.

[4] Wang, D. & Arbib, M. 1993. Timing and chunking in processing temporal order. IEEE Transactions on Systems, Man, and Cybernetics, 23(4), 993-1009.

[5] Maskara, A. & Noetzel, A. 1993. Training auto-associative recurrent neural network with preprocessed training data. Proceedings of the SPIE – The International Society for Optical Engineering: Science of Artificial Neural Networks II, 1966 (pp. 420-428). Orlando, FL.

[6] Liu, A. & Pentland, A. 1997. Towards real-time recognition of driver intentions. Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems (pp. 236-241). Boston MA.

[7] Narendra, K. S., Balakrishnan, J., & Ciliz, M. K. 1995. Adaptation and learning using multiple models, switching, and tuning. IEEE Control Systems Magazine, 15(3), 37-51.

[8] Pentland, A. & Liu, A. 1999. Modeling and prediction of human behavior. Neural Computation, 11, 229-242.

[9] Austin, K. B. & Rose, G. M. 1997. Automated behavior recognition using continuous-wave Doppler radar and neural networks. Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society: Magnificent Milestones and Emerging Opportunities in Medical Engineering, 4, 1997, Chicago, IL, October 30 to November 2, pp. 1458-1461.

[10] Weng, J. J. & Hwang, W. 1998. Toward automation of learning: the state self-organization problem for a face recognizer. Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, 1 (pp. 384-389). Nara, Japan.

[11] Strohal, M. & Onken, R. 1998. Intent and error recognition as part of a knowledge-based cockpit assistant. In S. K. Rodgers, D. B. Fogel, J. C. Bezdek & B. Bosacchi (Eds.), Applications and Science of Computational Intelligence: Proceedings of the SPIE, 3390, 287-299.

[12] Lesh, N., Rich, C. and Sidner, C. L. 1999. Using plan recognition in human-computer collaboration. In J. Kay (Ed.), Proceedings of UM99: Seventh International Conference on User Modeling (pp. 23-32). Wien, Austria: Springer.

[13] Wobke, W. 2002. "Two Logical Theories of Plan Recognition", Journal of Logic and Computation, 12(3), pp. 371-412, June 2002.

[14] Jiang, Y. F. and Ma, N. 2002. "Plan Recognition Algorithm based on Plan Knowledge Graph", Journal of Software, 13(4), pp. 686-692, April, 2002.

[15] Patterson, D., Liao, L., Fox, D., and Kautz, H. 2003. "Inferring High Level Behaviors from Low Level Sensors", Fifth Annual Conference on Ubiquitous Computing (UBICOMP 2003), Seattle, WA.

[16] Intille, S. S. and Bobick, A. F. 2001. "Recognizing Planned Multi-person Action", Computer Vision and Image Understanding, 81(3), pp. 414-445.

[17] Kerkez, B. and Cox, M. T. 2002. "Local Predictions for Case-based Plan Recognition", Proceedings of the 2002 European Conference on Case-Based Reasoning, pp. 189-203.

[18] Charniak, E. and Goldman, R. P. 1993. "A Bayesian Model of Plan Recognition", Artificial Intelligence, 64(1), pp. 53-79.

[19] Tambe, M. 1996. "Tracking Dynamic Team Activity", Proceedings of the National Conference on Artificial Intelligence (AAAI), 1996.

[20] Kaminka, G. A and Tambe, M. 2000. "Robust Multi-agent Teams via Socially-attentive Monitoring", Journal of Artificial Intelligence Research, 12, pp. 105-147.
[21] Kaminka, G. A., Pynadath, D. V, and Tambe, M. 2002. Monitoring Teams by Overhearing: A Multi-Agent Plan Recognition Approach", Journal of Artficial Intelligence Research, 1(124).
[22] Huber, M. J. 1996. "Plan-based Plan Recognition for Effective Coordination of Agents Through Observation", PhD dissertation, U. of Michigan.
[23] Han, K. and Veloso, M. 1999. "Automated Robot Behavior Recognition Applied to Robotic Soccer", Proc. of IJCAI-1999 Workshop on Team Behavior and Plan Recognition, 1999.
[24] Pynadath, D. V. and Wellman, M. P. 2000. "Probabilistic State-Dependent Grammars for Plan Recognition", Proc. of UAI-2000, pp. 507-514.
[25] Devaney, M. and Ram, A. 1998. "Needles in a Haystack: Plan Recognition in Large Spatial Domains involving Multiple Agents", Proceedings of the 15th national Conference on Artificial Intelligence, pp 942-947.
[26] Goldman, R. P., Geib, C. W. and Miller, C. A. 1999. "A New Model of Plan Recognition", Proc. of the Conf. on Uncertainty in Artificial Intelligence, 1999.
[27] Drewes, P. J., Gonzalez, A. J. and Gerber, W. 2000. "Interpreting Trainee Intent in Real Time in a Simulation-based Training System", Transactions of the Society for Computer Simulation, Vol. 17, No. 3, September 2000, pp. 120-134.
[28] Turner, R. M. 1994. Adaptive reasoning for real-world problems: a schema based approach. Hillsdale, NJ: Lawrence Erlbaum Associates.
[29] Turner, R. M. 1998. Context-mediated behavior for intelligent agents. In B. R. Gaines (Ed.), International Journal of Human-Computer Studies: Incorporating Knowledge Acquisition, 48(3), 307-330.
[30] Brezillon, P. 2004. "Representation of Procedures and Practices in Contextual Graphs", The Knowledge Engineering Review.
[31] Bass, E. J., Zenyuh, J.P., Small, R.L. and Fortin, S.T. 1996. "A Context-based Approach to Training Situation Awareness", Proceedings of the Third Annual Symposium on Human Interaction with Complex Systems, Los Alamitos, CA: IEEE Computer Society Press, pp. 89-95
[32] Gerber, W. J. 2001. Real-Time Synchronization of Behavioral Models With Human Performance in a Simulation. Doctoral Dissertation, University of Central Florida, Orlando.
[33] Drewes, P. J. 1997. Automated Student Performance Monitoring in Training Simulation. Doctoral Dissertation, University of Central Florida, Orlando.
[34] Hertz, J. A., Krough, A. S. & Palmer, R. G. 1991. Introduction to the Theory of Neural Computation. Redwood City, CA: Addison-Wesley Publishing Company.
[37] Gerber, W. J. and Gonzalez, A. J. 2001. Behavior recognition results for behavioral vehicle model synchronization in distributed simulations. Proceedings of the 2001 Interservice/Industry Training, Simulation and Education Conference (pp. 260-270). Orlando, FL.
[35] Zadeh, L. A. 1987. Fuzzy sets. In R. R. Yager, S Ovchinnikov, R. M. Tong & H. T. Nguyen (Eds.), Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh (pp. 29-44). New York: John Wiley & Sons, Inc. (Reprinted from Information and Control, 8, 338-353, 1965, New York: Academic Press)
[36] Langari, R. & Yen, J. 1995. Introduction to fuzzy logic control. In J. Yen, R Langari & L A. Zadeh (Eds.), Industrial Applications of Fuzzy Logic and Intelligent Systems (pp. 3-39). Piscataway, NJ: IEEE Press.