Self-Organizing Agents for Reinforcement Learning in Virtual Worlds

Yilin Kang, *Student Member, IEEE* and Ah-Hwee Tan, *Senior Member, IEEE*

***Abstract*— We present a self-organizing neural model for creating intelligent learning agents in virtual worlds. As agents in a virtual world roam, interact and socialize with users and other agents as in real world without explicit goals and teachers, learning in virtual world presents many challenges not found in typical machine learning benchmarks. In this paper, we highlight the unique issues and challenges of building learning agents in virtual world using reinforcement learning. Specifically, a self-organizing neural model, named TD-FALCON (Temporal Difference - Fusion Architecture for Learning and Cognition), is deployed, which enables an autonomous agent to adapt and function in a dynamic environment with immediate as well as delayed evaluative feedback signals. We have implemented and evaluated TD-FALCON agents as virtual tour guides in a virtual world environment. Our experimental results show that the agents are able to adapt and improve their performance in real time. To the best of our knowledge, this is one of the few in-depth works on building complete learning agents that adapt their behaviors through real time reinforcement learning in virtual world.**

I. Introduction

Virtual world has become a popular platform used in a variety of contexts, including teaching in classrooms, informal learning, distance learning, business, and e-commerce. Studies in South Korea have recently shown that users prefer virtual world to television [1]. Gartner even predicted that 80 percent of the Internet users will be actively participating in non-gaming virtual world by the end of 2011.

With the popularity of virtual world, many people have been working on various approaches to incorporate intelligent learning agents to improve its interactivity and playability. Indeed, learning in a virtual world, just like in the real world, poses many challenges not addressed by traditional machine learning algorithms. In particular, learning in virtual world is typically unsupervised, without an explicit teacher to guide the agent in learning. Furthermore, it requires an interplay of a myriad of learning paradigms.

In this paper, we highlight the issues and challenges of building learning agents in virtual world using reinforcement learning. Specifically, we investigate how a self-organizing neural model, known as TD-FALCON [2], may be exploited as learning agents in virtual world. TD-FALCON integrates temporal difference methods [3] and self-organizing neural networks [4], [5] for reinforcement learning with delayed evaluative feedback. Recently applied to a first-person shooter game, namely Unreal Tournament 2004 [6], the TD-FALCON agent has been shown to be capable of adjusting its tactical strategies and choosing appropriate weapons during run time similar to a human player. In this work, we extended TD-FALCON in a non-trivial way to address the challenges of learning in virtual world, including the lack of explicit goals and well defined evaluative feedback.

By incorporating TD-FALCON, an agent will be able to learn from sensory and evaluative feedback signals received from the virtual environment without involving human supervision and intervention. In this way, the agent needs neither an explicit teacher nor a perfect model to learn from. Performing reinforcement learning in real time, it is also able to adapt itself to the variations in the virtual environment and changes in the user behavior patterns. Furthermore, by incorporating temporal difference learning, TD-FALCON agents can overcome the issues, such as the absence of immediate reward (or penalty) signals in virtual world by estimating the credit of an action based on what it will lead to eventually.

We have developed personal agents using TD-FALCON and deployed them in a 3-D virtual world called Youth Olympic Games (YOG) Co-Space. In this application, the personal agents are designed to befriend human users and proactively offer them with functions and services in the virtual environment. Empirical experiments based on synthetic user models and real users both supported the validity of our approach. To the best of our knowledge, this is one of the few in-depth works on building complete agents that perform reinforcement learning in real time and adapt their actions and behaviors with immediate as well as delayed evaluative feedback in virtual world.

The rest of this paper is organized as follows. In section II, we give a brief review of the related work. In section III, we discuss the issues and challenges of building learning agents in virtual world. We provide an overview of TD-FALCON and a discussion of its key enabling features in section IV. In section V, we present the procedure and methods for using TD-FALCON for reinforcement learning in real time. In section VI, we describe the case study on the YOG Co-Space with the results of the empirical experiments and real user case study. The final section concludes and highlights future work.

II. Related Work

Intelligent agents have been popularly used for improving the interactivity and playability of virtual worlds. However, most such agents are based on scripts or predefined rules. For example, in the Virtual Theater project, synthetic actors portray fictive characters and provide improvising behaviors.

The authors are with the School of Computer Engineering , Nanyang Technological University, Singapore 639798, Singapore (email: kang0028@ntu.edu.sg; asahtan@ntu.edu.sg).

The agents are based on a scripted social-psychological model with the defined personality traits which rely on the values of moods and attitudes [7]. Agents in Metaverse, built using Active Worlds, are capable of performing the tasks typically associated with human beings, such as taking tickets for rides and acting as shopkeepers. However, these agents are basically reactive agents which work in a hard-coded manner. Virtual psychotherapist ELIZA [8], designed to take care of the 'patients', is also achieved with rule-based, adeptly modeled small talk. A conversational virtual agent Max has been developed as a guide to the HNF computer museum, where he interacts with visitors and provides them with information daily [9]. However, the design remains rule-based.

In view of the limitations of static agents, some researchers have adopted learning methods into service agents in virtual world. For example, Yoon et.al. present a Creature Kernel framework to build interactive synthetic characters in the project Sydney K9.0 [10]. Their agents can reflect the characters' past experience and allow individual personalization. But all the capabilities of the agents rely on past knowledge and couldn't adapt to user gradually during run time. The co-present agents in a virtual gallery [11] utilize a knowledge base containing general input response knowledge, augmented with knowledge modules for special domains. More recently, an embodied conversational agent that serves as a virtual tour guide in Second Life has been implemented by Jan [12]. It uses NPCEditor [13] to learn the best output for any input from a training set of linked questions and answers. Again, it learns from past experience but does not adapt over time according to the habits of a particular player or the changes in the environment.

All the work described above have developed a wide range of agents in virtual world with specific motivations. However, to the best of our knowledge, there has been very few in-depth work, if any, building complete agents that perform reinforcement learning and adapt their actions and behaviors in real time. Our work is motivated by such a consideration.

III. Issues and Challenges

Learning in virtual world presents many challenges not found in typical machine learning benchmark problems. The key issues are discussed as follows.

A. What Are The Goals?

Traditional games, such as card games and board games, often have a clear objective of defeating the adversary. In first-person shooter games, the primary concern is to survive and kill the enemies. For a real-time strategy game, the goal is to secure the team and destroy the opponents' assets. In contrast, players in virtual world explore the environment, meet other residents, socialize, participate in activities, create and trade virtual properties and services with one another without explicit goals. This key characteristic has led to an important distinction between virtual world and traditional games on the requirement of learning.

The issue of implicit goals also brings about other issues such as how to measure the learning performance. As the players in the virtual world may roam in the virtual world without any keen desire, how we can measure the learning performance is one of the main concerns.

B. When to Start and Stop Learning?

Exploring and playing in a virtual world is a continuous experience. Depending on the constraints and implementation of the virtual world, sensory signals and evaluative feedback are received and updated in synchronous or asynchronous fashion. It is thus important to associate an evaluative feedback to the correct state and action for reinforcement learning. Furthermore, agents need not learn all the time, as an agent may be left idle at times while the players wonder around or engage in chatting with other online residents. Enabling learning at all time consumes excessive computing resources and results in low efficiency.

C. How to Compute Evaluative Feedback?

The most important feature distinguishing reinforcement learning from other types of learning is that it uses training information that evaluates the actions taken rather than the instructs of correct actions [3]. It requests for active exploration as well, which includes an explicit trial-and-error search for good behavior [14]. In standard reinforcement learning tasks, the outcome of a trial is typically used in computing the evaluative feedback. In a First Person Shooting game, for example, a reward of 0.5 can be given for a hit and a reward of 1 is given for a kill during combat [6]. As virtual world is a continuous experience, evaluative feedback in virtual world cannot be defined simply by the final outcome of a game play. Since an agent is supposed to continue assisting or servicing a user, a more subtle way of defining evaluative feedback should be considered.

D. How to Learn in A Dynamic, Real-time Environment?

Virtual world presents a highly dynamic environment. In particular, it is a system driven by a rendering cycle that ideally works at 50 or 60 Hertz so that changes appear as smooth animation [15]. Therefore, agents in virtual world cannot afford to operate in terms of primitive actions, such as "moves one step to north" or "turns to face northwest". Instead, abstractions have to be formed, which allow learning to conduct forward searches in a manageable abstract space and to translate the solutions found back into action sequences in the original state space. In addition, learning should be made online in order to meet the real-time needs.

IV. TD-FALCON as Virtual Agents

TD-FALCON [2], [16] incorporates Temporal Difference (TD) methods to estimate and learn value functions of action-state pairs $Q(s, a)$ that indicates the goodness for a learning system to take a certain action a in a given state s. Such value functions are then used in the action selection mechanism, also known as the *policy*, to select an action with the maximal payoff. TD-FALCON algorithm [2] selects an action with the

maximal Q-value in a state s by enumerating and evaluating each available action a by presenting the corresponding state and action vectors **S** and **A** to FALCON.

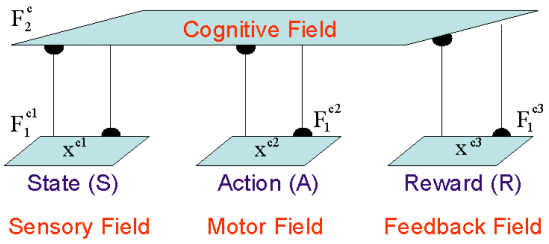

Fig. 1. TD-FALCON architecture.

There are several considerations making TD-FALCON a suitable candidate for building learning agents in virtual world. We provide a discussion of the key enabling properties below.

1) **Self-Adaptation:** TD-FALCON learns by getting rewards, which does not involve any human supervision and intervention. In this way, the agent does not need a perfect model to learn from. The agent is able to adjust its behaviors in different situations, if enough training is provided.

2) **Generalization:** Memory consumption is a major limiting factor to the size of the problems that a learning agent can tackle. Classical approaches of reinforcement learning usually causes a scalability issue for continuous and/or very large state and action spaces since policy function and/or value function must be learned for each and every possible state or possible pair of state and action. A self-organizing neural network can ease the memory requirement by its ability of generalizing and finding patterns in data. By incorporating self-organizing neural network, TD-FALCON can deal with the problem of learning a massive amount of data in virtual environment.

3) **TD Learning:** It is important to note that reward information may not always be available in a virtual world environment. TD-FALCON has incorporated temporal difference methods to estimate and learn value function of state-actions pairs $Q(s, a)$ based on a combination of current and future estimated rewards. By making use of TD methods, they can perform multiple-step prediction in virtual world, in which the merit of an action can only be known after several steps into the future.

4) **Fast and Stable Real-time Learning:** Based on Adaptive Resonance Theory (ART), TD-FALCON is capable of performing fast and stable reinforcement learning in real time. Indeed, it has shown in a prior study that TD-FALCON produces a stable performance in a pace much faster than those of standard gradient-descent based reinforcement learning systems [2].

V. Learning Algorithm

A. FALCON Dynamic

FALCON employs a three-channel architecture (Figure 1), comprising a category field F_2 and three input fields, namely, a sensory field F_1^{c1} for representing current states, a motor field F_1^{c2} for representing actions, and a feedback field F_1^{c3} for representing reward values [17]. The generic network dynamics of FALCON, based on fuzzy ART operations [18], is described as follows.

Input vectors: Let $\mathbf{S} = (s_1, s_2, \ldots, s_n)$ denote the state vector, where $s_i \in [0, 1]$ indicates the value of sensory input i. Let $\mathbf{A} = (a_1, a_2, \ldots, a_m)$ denote the action vector, where $a_i \in [0, 1]$ indicates the preference of a possible action i. Let $\mathbf{R} = (r, \overline{r})$ denote the reward vector, where $r \in [0, 1]$ is the reward signal value and $\overline{r}$ is given by $\overline{r} = 1 - r$. The whole input vectors are with complement coding.

Activity vectors: Let $\mathbf{x}^{ck}$ denote the F_1^{ck} activity vector for $k = 1, \ldots, 3$. Let $\mathbf{y}$ denote the F_2 activity vector.

Weight vectors: Let $\mathbf{w}_j^{ck}$ denote the weight vector associated with the jth node in F_2 for learning the input patterns in F_1^{ck} for $k = 1, \ldots, 3$. Initially, F_2 contains only one *uncommitted* node. An *uncommitted* node is one which has not been used to encode any pattern and its weight vector contains all 1s.

Parameters: The FALCON's dynamics is determined by choice parameters $\alpha^{ck} \geq 0$, learning rate parameters $\beta^{ck} \in [0, 1]$, contribution parameters $\gamma^{ck} \in [0, 1]$ and vigilance parameters $\rho^{ck} \in [0, 1]$ for $k = 1, \ldots, 3$.

The FALCON pattern processing cycle comprises of five key stages, namely code activation, code competition, activity readout, template matching, and template learning, as described below.

Code activation: A bottom-up propagation process first takes place in which the activities of the category nodes in the F_2 field are computed. Specifically, given the activity vectors $\mathbf{x}^{c1}, \mathbf{x}^{c2}, \mathbf{x}^{c3}$, for each F_2 node j, the choice function T_j is computed as follows:

$$T_j = \sum_{k=1}^{3} \gamma^{ck} \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_j^{ck}|}{\alpha^{ck} + |\mathbf{w}_j^{ck}|}, \tag{1}$$

where the fuzzy AND operation $\wedge$ is defined by $(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i)$, and the norm $|.|$ is defined by $|\mathbf{p}| \equiv \sum_i p_i$ for vectors $\mathbf{p}$ and $\mathbf{q}$.

Code competition: A code competition process follows under which the F_2 node with the highest choice function value is identified. The system is said to make a choice when at most one F_2 node can become active after the code competition process. The winner is indexed at J where $T_J = \max\{T_J : \text{for all } F_2 \text{ node } j\}$.

When a category choice is made at node J, $y_J = 1$ and $y_j = 0$ for all $j \neq J$. This indicates a winner-take-all strategy.

Activity readout: The chosen F_2 node J performs a readout of its weight vectors into the input fields F_1^{ck} such that

$$x^{ck(new)} = x^{ck(old)} \wedge w_J^{ck}. \tag{2}$$

The resultant F_1^{ck} activity vectors are thus the fuzzy AND of their original values and their corresponding weight vectors.

Template matching: Before the node J can be used for learning, a template matching process checks that the weight templates of node J are sufficiently close to their respective

input patterns. Specifically, resonance occurs if for each channel k, the *match function* m_J^{ck} of the chosen node J meets its vigilance criterion ρ^{ck}:

$$m_J^{ck} = \frac{|\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck}|}{|\mathbf{x}^{ck}|} \geq \rho^{ck}. \quad (3)$$

If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function T_J is set to 0 for the duration of the input presentation. The search process then selects another F_2 node J until a resonance is achieved.

Template learning: Once a resonance occurs, for each channel ck, the weight vector $\mathbf{w}_J^{ck}$ is modified by the following learning rule:

$$\mathbf{w}_J^{ck(\text{new})} = (1 - \beta^{ck})\mathbf{w}_J^{ck(\text{old})} + \beta^{ck}(\mathbf{x}^{ck} \wedge \mathbf{w}_J^{ck(\text{old})}). \quad (4)$$

When an uncommitted node is selected for learning, it becomes *committed* and a new uncommitted node is added to the F_2 field. Fusion ART thus expands its network architecture dynamically in response to the input patterns.

B. TD-FALCON

TD-FALCON integrates temporal difference learning with the FALCON dynamics described above for reinforcement learning. The general sense-act-learn algorithm is summarized in Table I.

As mentioned earlier, agents do not learn all the time in the virtual world. Therefore, TD-FALCON first needs to detect an opportunity for learning. The cues for learning typically include a change in the situation (state) and a receipt of an evaluative feedback. Given the current state s, the FALCON network is used to predict the value of performing each available action a in the action set $\mathcal{A}$ based on the corresponding state vector $\mathbf{S}$ and action vector $\mathbf{A}$. The value functions are then processed by an action selection strategy (also known as policy) to select an action.

After performing the action, a feedback (if any) from the environment is received. In the virtual worlds, the feedback could be a synthesis of the user's verbal expression, body language or emotional expression. Upon receiving the feedback, a TD-formula is used to compute a new estimate of the Q-value for performing the chosen action in the current state.

A typical Temporal Difference (TD) equation for iterative estimation of the value functions Q(s,a) is given by

$$\Delta Q(s,a) = \alpha TD_{err}, \quad (5)$$

where $\alpha \in [0,1]$ is the learning parameter and TD_{err} is a function of the current Q-value predicted by FALCON and the Q-value newly computed by the TD formula.

Using the Q-learning rule, the temporal error term is computed by

$$TD_{err} = r + \gamma \text{max}_{a'} Q(s',a') - Q(s,a), \quad (6)$$

where r is the immediate reward value, $\gamma \in [0,1]$ is the discount parameter, and $\text{max}_{a'} Q(s',a')$ denotes the maximum estimated value of the next state s'.

This new Q-value is then used as the teaching signal (represented as reward vector $\mathbf{R}$) for FALCON to learn the association of the current state and the chosen action to the estimated value. The above steps repeat until a terminal state s is reached. This happens when the user has completed his/her journey of exploring the virtual world.

VI. A Case Study on YOG Co-Space

Co-Spaces are virtual worlds developed for mirroring a real physical world in terms of look-and-feel, functions and services. The objective of Youth Olympic Games (YOG) Co-Space is to introduce the YOG and the hosting country to visitors around the world in an interactive and playable manner. To achieve this objective, we are in the process of developing and populating human-like cognitive agents in the form of autonomous avatars that roam in the landscape of YOG Co-Space. The agents are designed to be aware of its surrounding and can interact with users through their human avatars. With the autonomous avatars befriending and providing personalized context-aware services to human avatars, we aim to make the content and services readily available to the users.

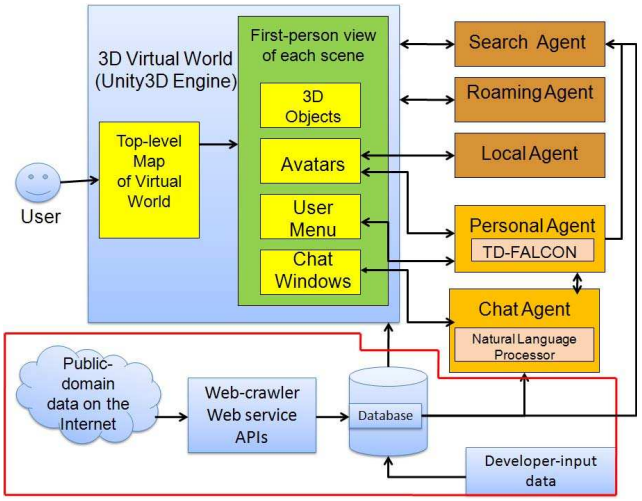

Fig. 2. The architecture of Co-Space.

The architecture of Co-Space is shown in Figure 2. As illustrated in this framework, the TD-FALCON based personal agent works in conjunction with the search agent in recommending functions and services to the users. Specifically, the personal agent determines the appropriate type of services to recommend whereas the search agent retrieves the specific services based on the environment situations as well as the users' context parameters. Figure 3 provides a screenshot of the virtual world, showing the personal agent Carol serving the user.

TABLE I

GENERIC FLOW OF THE TD-FALCON ALGORITHM

1. Initialize the FALCON network
2. Given the current state s, for each available action a in the action set $\mathcal{A}$, predict the value of the action Q(s,a) by presenting the corresponding state and action vectors $\mathbf{S}$ and $\mathbf{A}$ to FALCON.
3. Based on the value functions computed, select an action a from $\mathcal{A}$ following an action selection policy.
4. Perform the action a, observe the next state s', and receive a reward r (if any) from the environment.
5. Estimate the value function $Q(s, a)$ following a temporal difference formula given by $\Delta Q(s, a) = \alpha T D_{err}$
6. Present the corresponding state, action, and reward (Q-value) vector, namely $\mathbf{S}$, $\mathbf{A}$ and $\mathbf{R}$, to FALCON for learning.
7. Update the current state by $s = s'$.
8. Repeat from Step2 until s is a terminal state.

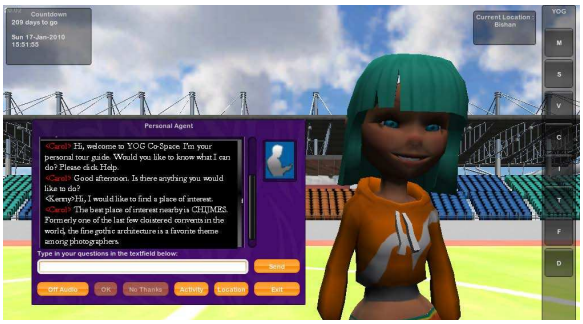

Fig. 3. A screenshot of TD-FALCON based personal agent in Co-Space.

A. Embodiment of TD-FALCON in Co-Space

1) State Representation: The input state of the personal agent consists of five sets of attributes, namely time, location and the player's request, interest and current activity. A summary of these attributes together with the possible values is given in Table II. All attributes adopt a binary encoding scheme in the state vector. Although we include user request as part of the state space, the agents are intended to work without explicit user request. In other words, they are supposed to proactively make recommendation based on user's interest, context and situation.

TABLE II

ATTRIBUTES AND POSSIBLE VALUES IN THE STATE SPACE

Attributes	Values
Time	Morning, lunch Time, Afternoon, Dinner Time, Evening, Bed Time
Player's location	Central, North, South, East, West
Player's request	Hotel, Dining, YOG Venue, Place of Interest(POI), Shopping, Unknown
Player's interest	Dining, YOG, POI, Shopping, Unknown
Player's activity	Hotel, Dining, YOG, POI, Shopping, Unknown

2) Action Space Representation: The personal agent is designed to determine the most appropriate service for recommendation to its user. The action field thus consists of recommendations of five types of services, namely hotel, dinning, YOG, place of interest and shopping. Using a binary encoding scheme, the action vector is represented as $\mathbf{a} = (a_1, a_2, \ldots, a_5)$, where $a_j = 1$ and $a_k = 0$ for all $k \neq j$ indicate that action j is recommended.

3) Computing Reward Signals: The reward function $r(t)$ is defined as a synthesis of the player's feedback and mood as $r(t) = (f(t) + m(t))/2$, where f is an explicit reward based on the user feedback through the dialog menu and f is an implicit reward value suggested by the gestures of the user. The dialog menu enables a player to choose among "wonderful", "thanks", "good", "fair", and "leave me alone", which correspond to a reward value of 1, 0.75, 0.5, 0.25 and 0 respectively. Meanwhile, we gauge a player's mood by observing the gestures of the player avatar. Gestures, such as waving, dancing, and jumping, indicate a positive reward of 1 since they indicate satisfaction with the service. In contrast, a reward of 0 is given to gestures, such as angry and walk away, which suggest unhappiness.

B. Empirical Experiments

We build a TD-FALCON network comprising 33 nodes in the sensory field, five nodes in the action field, and two nodes in the reward field. We conduct four sets of experiments to analyze how TD-FALCON performs in various settings, including learning in real time, learning with pre-inserted rules, adapting to changing user models, and learning with delayed feedback. In order to evaluate our system performance empirically, we design an automatic test procedure using synthetic user models. We simulate the user's feedback to the agent's service by using a set of user logic. By matching the agent's recommendations with the user's expectation, we are able to compute reward signals that can be used to guide the agent in learning. A sample set of user logic is given in Table III as an illustration.

1) Performance Measures: As the main role of personal agents is to provide services to the users, one important performance indicator is the accuracy of the service given to the users over a period of time. In addition, in the motivation of elevating the user experience in virtual world, we define a general goal of maximizing the user's satisfaction. For a recency-biased indication of the system's performance, we compute the user's satisfaction index at time t by $S(t) = \gamma r(t) + (1 - \gamma)\, S\,(t - 1)$, where $S(0) = 0$, $\gamma \in [0, 1]$ is the recency weighting factor, and $r(t)$ indicates the reward value received at time t.

TABLE III

USER LOGIC FOR GENERATING REWARDS.

IF	Time=Lunch or Dinner Time, Activity != Dinning, Request = Unknown,
EXPECT	Recommend restaurant
IF	Time=Lunch or Dinner Time, Activity = Dinning, Request = Unknown, Interest contains X
EXPECT	Recommend X
IF	Time = Bed Time, Request = Unknown
EXPECT	Recommend Hotel

2) Learning with User Feedback: TD-FALCON uses a default set of parameter values as listed in Table IV. The recommendation accuracy without pre-inserted rules and user satisfaction index (γ=0.25) averaged at 100-trial intervals over 3000 trials are shown in Figure 4 and Figure 5 respectively. The results, obtained after averaging across five sets of experiments, indicate that the agents are able to gradually improve their performance through user feedback continuously over time. We have also conducted experiments with different sets of vigilance parameter values. As summarized in Table V, vigilance parameters have a considerable influence on the system: a higher vigilance produces highly detailed memories, while a lower vigilance results in more general memories. Nevertheless, we see that the performance of the agents are generally stable over a reasonable range of parameter values. Reducing the vigilance values drastically however causes a degrade in the system's performance.

TABLE IV

TD-FALCON PARAMETER SETTING.

FALCON Parameters	Values
Choice parameter(α^{c1},α^{c2},α^{c3})	0.1, 0.1, 0.1
Learning rates(β^{c1},β^{c2},β^{c3})	1.0, 1.0, 1.0
Contribution parameter(γ^{c1},γ^{c2},γ^{c3})	1/3, 1/3, 1/3
Baseline vigilance parameter(ρ^{c1},ρ^{c2},ρ^{c3})	1.0, 1.0, 1.0
TD Learning Parameters	
TD learning rate α	1
Discount factor γ	0
Initial Q-value	0
ϵ-greedy Action Policy Parameters	
Initial ϵ value	0.5
ϵ decay rate	0.0002

TABLE V

PERFORMANCE OF TD-FALCON WITH DIFFERENT PARAMETER VALUE SETTINGS.

ρ^{c1}, ρ^{c2}, ρ^{c3}	Success rate at 3000 decision cycles	Number of Nodes
1.0 1.0 1.0	97%	523
1.0 1.0 0.5	99%	515
0.9 0.2 0.5	98%	518
0.8 0.2 1.0	78%	178

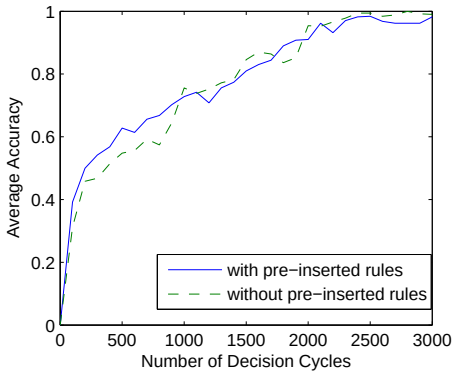

Fig. 4. Recommendation accuracy of TD-FALCON with and without pre-inserted rules.

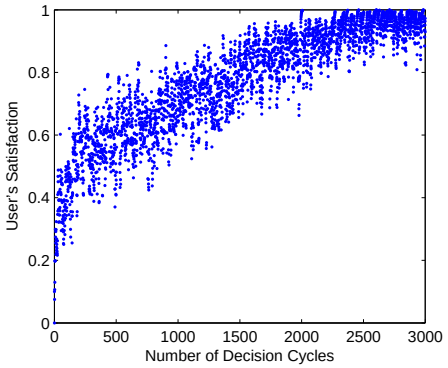

Fig. 5. User satisfaction index obtained over time.

3) Experimenting with Pre-existing Rules: In this set of the experiments, we investigate the performance of TD-FALCON with the pre-existing knowledge in the form of symbolic rules as illustrated in Table VI. A total of 180 rules are generated based on a selected user model. Each of these rules is inserted into TD-FALCON with a reward value of 1 before learning commences.

TABLE VI

A SAMPLE SET OF PRE-INSERTED RULES.

IF	Time = Lunch Time or Dinner Time, Request = Unknown, Interest = POI and YOG, Activity = Hotel
THEN	Action = Recommend Dining
IF	Time = Bed Time, Request = Unknown, Interest = POI and YOG, Activity = Hotel
THEN	Action = Recommend Hotel
IF	Time = Morning or Afternoon or Evening, Request = Unknown, Interest = POI and YOG, Activity = Hotel;
THEN	Action = Recommend YOG or POI

Comparing to the 33,480 possible combinations of state and actions, the 180 rules represent only a small fraction of the knowledge. Figure 4 and Figure 6 depict the comparative performance of TD-FALCON with pre-inserted rules

and without pre-inserted rules in terms of recommendation accuracy and user satisfaction index respectively. We can observe that with the pre-inserted rules, the performance generally increases slightly faster in the first few hundreds of the interaction cycles comparing to that without pre-inserted rules. However, their performance are largely similar after 1000 interaction cycles.

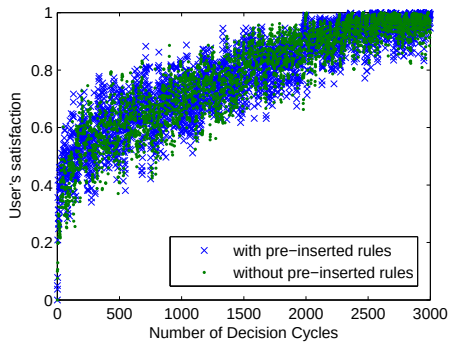

Fig. 6. User satisfaction index with and without pre-inserted rules.

4) Adapting to Changing User Logic: In this set of the experiments, we evolve a user model during run-time to investigate how the agents may cope with a shift in user's interests. We changed 10% of the rules after 3000 learning trials to shift the user's interest to dinning and shopping.

To put the adaptation performance of TD-FALCON into perspective, we build an agent based on a set of rules, which readily achieves 100% accuracy at the beginning of the experiments. As shown in Figure 7 and Figure 8, the accuracy of the rule based system (RBS) suffers a significant drop and hovers around 60% after the change. This is because when the user model is changed, it does not have the capability of adapting to it. In contrast, TD-FALCON, despite suffering a drop to an accuracy of 55%, quickly adapts to the new user. After 1000 decision cycles, it manages to regain an accuracy of more than 90%.

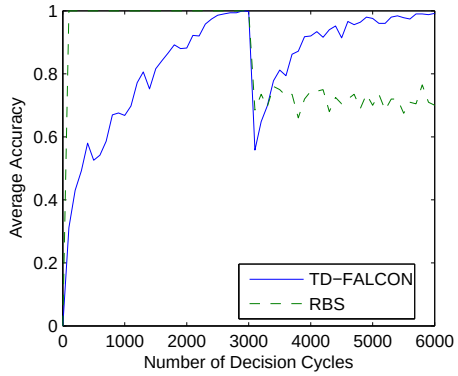

Fig. 7. Recommendation accuracy of TD-FALCON and the rule based agent (RBS).

5) Learning with Delayed Feedback: It is significant to note that all the above mentioned experiments rely on the

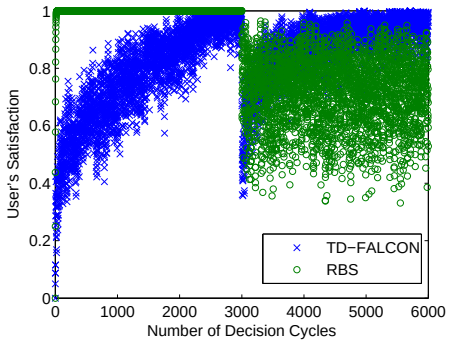

Fig. 8. User satisfaction index obtained by TD-FALCON and the rule based system (RBS).

feedback obtained after performing each action. However, in a realistic environment, it may take a long sequence of actions before a reward or penalty is finally given. This is the reason why we incorporate TD method to estimate the credit of an action based on what it will lead to eventually rather than learning a function mapping states to actions directly.

In order to observe the effect of temporal difference learning, we evaluate TD-FALCON with a set of synthetic users, who may or may not provide feedback to each and every recommendations given by the agents. As a guide, the users respond to roughly 50% of the recommendations given. For the rest of the cases, the immediate reward value is zero and the agents rely on the Q-value computed using the TD-learning rule. The performance of TD-FALCON, in terms of prediction accuracy and user satisfaction index, can be found in Figure 9 and Figure 10 respectively.

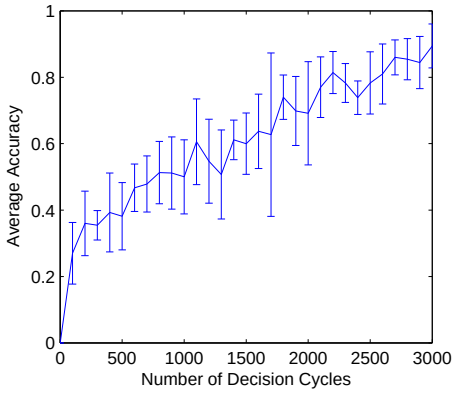

Fig. 9. Average recommendation accuracy (with range bars) of TD-FALCON agents without immediate rewards.

We note that learning with delayed feedback is a much more difficult task due to the lack of explicit immediate reward signals and the large variety of the input instances. For both performance measures, we observe that the overall performance is degraded comparing with those obtained in the previous experiments. However, a reasonable level of performance is still maintained as TD-FALCON manages the conflicting requirement of short-term reactivity and long-

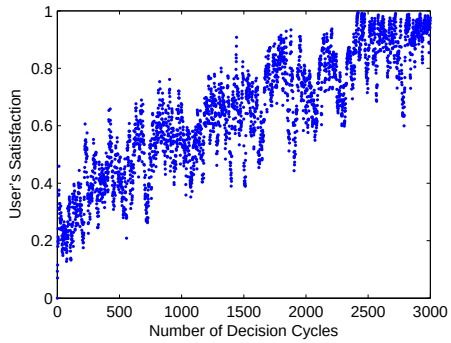

Fig. 10. User satisfaction index of TD-FALCON agents without immediate rewards.

term goal-driven behavior.

C. User case study

We invited several people to be the trial users of YOG Co-Space. These people have different interest profiles and different personal attitudes towards giving feedback rewards to system (Active/ Passive). We inserted 300 rules initially and let users try out the system. These rules represent some common knowledge of human-being. The criterion for measuring the performance is the success rate of recommending services.

TABLE VII
RESULT OF REAL USER STUDY.

userID	Interest profile	Active/ passive	Success rate	Number of Nodes
1	Sport	Active	90%	515
2	Dinning Shopping	Active	86%	518
3	POI Sport	Passive	83%	522
4	Shopping POI Dinning	Active	82%	558
5	Sport Shopping Dinning	Passive	77%	440

As shown in Table VII, although facing users with different interests and attitudes, our learning agent still achieved a good success rate of 83.6% on average. For people whose interest is relatively simple and active in providing feedback, the success rates are correspondingly higher than the others. As a matter of fact, this performance pattern is in accordance with that of a human tour guide as well.

VII. CONCLUSIONS

We set off to improve the interactivity and playability of virtual world and make the environment more enjoyable by employing agent technologies. In this paper, we have focused on the issues and challenges for building learning agents in virtual world. To this end, we presented TD-FALCON, which enables an autonomous agent to adapt and function in a dynamic environment, as the model for implementing learning agents for personal services in virtual worlds.

Our experimental results and user case study have so far supported the validity of our agent systems. Moving forward, we wish to develop agents that can roam autonomously, discover knowledge and services on its own, and accumulate its knowledge and capabilities over time. We also want to extend the capability of TD-FALCON based agents to more complex tasks involving more situational factors and actions. Another possible extension is to provide planning of customized itinerary for individual players in the virtual world.

REFERENCES

[1] J. Weinstein and J. Myers, "Same principles apply to virtual world expansion as to china and other new markets," *Media Village*, vol. 11, 2006.

[2] A.-H. Tan, N. Lu, and D. Xiao, "Integrating Temporal Difference Methods and Self-Organizing Neural Networks for Reinforcement Learning with Delayed Evaluative Feedback," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 230–244, 2008.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An Introduction*. MIT Press, 1998.

[4] A.-H. Tan, "Self-organizing Neural Architecture for Reinforcement Learning," in *International Symposium on Neural Networks*, 2006, pp. 470–475.

[5] A.-H. Tan, G. A. Carpenter, and S. Grossberg, "Intelligence through Interaction: Towards a Unified Theory for Learning," in *Proceedings of International Symposium on Neural Networks*, 2007, pp. 1094–1103.

[6] D. Wang, B. Subagdja, A.-H. Tan, and G.-W. Ng, "Creating human-like autonomous players in real-time first person shooter computer games," in *proceedings, Twenty-First Annual Conference on Innovative Applications of Artificial Intelligence (IAAI'09)*, 2009, pp. 173–178.

[7] D. Rousseau and B. Hayes-Roth, "A social-psychological model for synthetic actors," *Stanford Knowledge Systems Laboratory Report KSL-97-07*, 1997.

[8] J. Weizenbaum, "ELIZA: a computer program for the study of natural language communication between men and machines," *Communications of the ACM*, vol. 9, 1996.

[9] S. Kopp, L. Gesellensetter, N. C. Krmer, and I. Wachsmuth, "A conversational agent as museum guide - design and evaluation of a real-world application," *Intelligent Virtual Agents*, pp. 329–343, 2005.

[10] S. Yoon, R. Burke, B. Blumberg, and G. Schneider, "Interactive training for synthetic characters," in *AAAI*, 2000, pp. 249–254.

[11] M. Gerhard, D. Moore, and D. Hobbs, "Embodiment and copresence in collaborative interfaces," *Int. J. Hum.-Comput. Stud.*, vol. 64(4), pp. 453–480, 2004.

[12] D. Jan, A. Roque, A. Leuski, J. Morie, and D. Traum, "A virtual tour guide for virtual worlds," in *Intelligent Virtual Agents Conference (IVA)*, 2009, pp. 372–378.

[13] A. Leuski and D. Traum, "A statistical approach for text processing in virtual humans." in *26th Army Science Conference*, 2008.

[14] I. Millington, *Artificial intelligence for games*. Morgan Kaufmann, 2006.

[15] R. Aylett and M. Luck, "Applying artificial intelligence to virtual reality: Intelligent virtual environments," *Applied Artificial Intelligence*, vol. 14(1), pp. 3–32, 2000.

[16] A.-H. Tan, "Direct Code Access in Self-Organizing Neural Networks for Reinforcement Learning," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2007, pp. 1071–1076.

[17] ——, "FALCON: A Fusion Architecture for Learning, COgnition, and Navigation," in *Proceedings of International Joint Conference on Neual Networks*, 2004, pp. 3297–3302.

[18] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks*, vol. 4, pp. 759–771, 1991.