On the Use of Intelligent Agents as Partners in Training Systems for Complex Tasks[1]

Thomas R. Ioerger, Joseph Sims, Richard A. Volz
Department of Computer Science, Texas A&M University
College Station, TX 77843-3112
{ioerger,jms3627,volz}@cs.tamu.edu

Judson Workman, Wayne L. Shebilske
Department of Psychology, Wright State University
Dayton, OH 45435-0001
judwwork@yahoo.com, wayne.shebilske@wright.edu

Abstract

Training protocols that involve working with a human partner have been shown to be beneficial for learning complex tasks. In this paper, we explore emulating the function of the partner with an intelligent agent. Given a cognitive task analysis, the task can be decomposed into cognitive components, and these behaviors can be independently automated using agent-programming techniques. Then a trainee and the agent can work together to solve practice problems, each taking responsibility for a different function. We argue that it is desirable not only for the agent to produce correct and consistent behavior (e.g. demonstrating the optimal strategy), but also to appear realistic (human-like, including errors), and we show how this can be achieved by introducing randomness in an agent's decisions. We implemented a Partner Agent for Space Fortress, a laboratory task designed to be representative of complex tasks, and found that trainees who swapped roles with this agent during training achieved significantly higher performance scores asymptotically than those who trained using a standard (whole-task) training protocol. We also simulated 3 different levels of expertise and found that trainees who worked with an "expert-level" agent received the most benefit.

Introduction

In the modern industrialized and technology-driven world, there is a great need for development of new training methods for complex tasks. By complex tasks, we mean tasks that have a cognitive dimension of difficulty, such as demands on reasoning, attention, memory, and so on, not just a physical skill based on strength or dexterity (though they may include these). Examples of complex tasks include driving vehicles, piloting aircraft, operating machinery, etc. Such tasks require extensive training for operators to learn the necessary details of how a device works, modes of operation, procedures for controlling it, regulations and limitations, signals for error/failure conditions, and means for recovering from them. Operators must not only learn this information by memory, but also be able to apply it in practice, often under real-time constraints with competing demands on perception, memory, etc. Schneider (Schneider, 1985) gives a characterization of complex tasks and discusses issues for training.

The need for new training systems is more prevalent today than ever before, especially given economic pressures to do more with fewer resources. In the commercial/manufacturing world, workers constantly need re-training to operate new models of machines as they are developed, or shift processes/jobs as the market demands. The military, too, is filled with jobs involving operation of complex, technological equipment, which people (typically young adults with little prior experience) must be trained rapidly to operate safely and effectively. In all these environments, the transition between textbook knowledge and practical skill must be achieved efficiently and effectively, minimizing both training resources (need for instructors, dedicated training equipment, simulators, etc.) and time.

Complex tasks present a fundamental challenge for the development of training systems. On the one hand, whole-task training (e.g. immersion, on-the-job training) is ineffective because the novice is usually over-whelmed by the complexity of the task. They fail at first, but they are often unable to comprehend why or make incremental improvements. On the other hand, part-task training (such as learning to steer a car and operate the pedals separately) can be less effective because novices do not get a chance to experience the inter-play between the parts. Part-task training does not allow trainees to practice performing the sub-tasks together, which often requires significant additional effort to manage shared cognitive resources, such as dividing or shifting attention.

One interesting training protocol that has shown promise for complex tasks is *partner-based training*, for example, the AIM (Active Interlocked Modeling) protocol (Shebilske et al., 1992). AIM involves groups of trainees working together to solve a problem (e.g. operating a device within a simulated scenario). In the case of AIM-Dyad, there are two trainees acting as partners. Each trainee performs part of the task while his partner does the other part, and later they switch roles. This potentially solves the dilemma of complex-task training because it reduces their individual demands, allowing them to focus on automating one cognitive component at a time, which is more tractable, while maintaining the context of the whole task.

It has been shown that the AIM protocol of partner-based training can improve performance over standard individual practice (e.g. whole-task training) for the same amount of time. For example, in Space Fortress, a computer-based laboratory task designed to emulate

[1]This work was supported in part by MURI grant #F49620-00-1-0326 from DoD and AFOSR.

characteristics of a complex task (specifically, flying a fighter jet), AIM-Dyad was implemented by alternatively having one trainee operate the joystick while the other trainee operated the mouse. Over a total of 10 hours of practice, trainees using this partner-based protocol performed as well at the task as individuals who had been trained using whole-task training. That is, they reached the same scores (individually) as a control group trained by simply giving them instructions and letting them try to maximize their score over the same amount of time (Shebilske et al., 1992; Arthur et al., 1997). Thus AIM-Dyad produces an increase in efficiency over the standard (individual) training protocol by reducing the time, equipment, and trainers needed for each group in half.

The fact that AIM can provide training that is equal to an individualized training method is somewhat surprising, given that each trainee experiences only half of the hands-on experience on average. The proposed explanation for this is that, while trainees are performing their own part of the task, they are "modeling" the behavior of their partner. In fact, the magnitude of performance improvement has been shown to correlate with intelligence measures of one's partner (Shebilske et al., 1999), suggesting that they are learning from each other more than just their own part of the task (Bandura, 1986).

However, this benefit of training with a partner does not intrinsically require interaction between the partners. In a previous study, the role of social variables (ranging from verbal communication to visual cues such as body language) were investigated by having the dyad perform the task in physically-separated cubicles with connected consoles. Still, they reached the same level of proficiency after training (Shebilske et al., 1999). Furthermore, this effect held even when trainees were told that the other part of the task was being performed by a computer.

This observation makes an important suggestion: that *the role of a partner in training could be automated by using an intelligent agent*. Agents are software programs that can autonomously make decisions and act to achieve goals in a dynamic (real-time) environment. There are many possible roles that agents could play in a training system. In this paper, we describe a principled approach to incorporating agents in training, called the Partner Agent protocol, based on the arguments for cognitive benefits given above. We discuss a number of design criteria and implementation issues in developing partner agents. To test our hypothesis about the effectiveness of this new protocol, we implemented partner agents for the Space Fortress task, and we show that trainees were able to out-perform (reach higher scores than) those trained with the standard control (whole-task) protocol.

The success of our first experiment led to an interesting follow-up question: *Does the level of expertise simulated by the partner agent affect the magnitude of performance improvement by the trainee*? Because trainees are believed to "model" the behavior of their partners, it might be expected that different behaviors by the agent will influence trainees differently. More specifically, if the agent's performance is similar to a novice (i.e. laden with errors), the trainee will not be able to learn correct behaviors, while if the agent's actions are too precise (i.e. expert), this might be incomprehensible to the novice, effectively setting an unattainable goal. Therefore, we ran a second experiment where we actively manipulated the level of expertise simulated by the partner agent, and we evaluated whether there was any difference in training with the different agents. The results show that training in Space Fortress with a partner agent that simulates a human expert provides the most benefit.

Partner Agent Design

There are a wide variety of ways in which intelligent agents could be used within a computer-based training system, ranging from automated opponents to coaches (Rickel and Johnson, 1999) to performance support. Intelligent agents are software programs that have the following characteristics: 1) they are *goal-oriented* (proactive, seeking to achieve goals given to them), 2) they are *reactive* (situated within a dynamic environment in which they need to take actions to change the state to achieve their goals), and 3) they are *autonomous* (can make decisions without human intervention) (Wooldridge and Jennings, 1995). In addition, agents may have other common characteristics, such as being *adaptive* (learning from their experiences) or *cooperative* (interacting with other agents or humans).

Intelligent agents have a potential for training that goes beyond traditional intelligent tutoring systems (ITS's) (Anderson et al., 1990). ITS's are systems designed for training based on AI techniques, especially expert systems and case-based reasoning. The trainee is usually presented with a problem to solve, and the ITS monitors the actions taken. If the problem was not solved correctly, an analysis of the actions is made to identify gaps in the trainee's knowledge, to be remediated by further instruction focussed in that area. The main challenge of an ITS is to interpret the actions the trainee takes and construct plausible explanations of the missing/incorrect knowledge that led to those actions. This is often called "user modeling," and can be performed by techniques ranging from abduction (proof completion) to plan recognition to probabilistic inference (e.g. with Bayesian belief nets).

While intelligent agents may incorporate user-modeling capabilities too, they go beyond ITS's by being able to dynamically interact with the problem-solving environment, and thus to drive (alter) the scenario, or actively participate with the trainee in solving the problem. An agent can be given goals that involve helping (facilitating) the trainee, correcting mistakes, reminding, off-loading tasks (performance support), providing advice on or explaining correct actions (decision aid), demonstrating correct behavior, adapting the scenario to the trainee's skill level (modifying events to be more or less challenging), making the scenario more realistic by simulating elements in the scenario reacting to the trainee's actions (more flexible than scripted scenarios), and even creating challenges by intentionally preventing

the trainee from succeeding by easy means and forcing them to apply deeper knowledge (e.g. a tactical enemy that is difficult to defeat in a combat simulation). These all require agent techniques such as planning and inference to be able to decide which actions to take to achieve their goals, given the current state of scenario (including user's prior actions).

Our approach, based on cognitive principles described above, is to use agents as active partners for trainees in solving problems. The advantage of using agents for creating virtual partners is that agents can be used to demonstrate correct behavior by giving them knowledge of the optimal strategy. Furthermore, agents will apply this knowledge consistently without getting tired, giving trainees a stable target behavior to model. This assumes that: a) sufficient inputs (perceptions) are available from the simulation environment to determine the correct behavior, b) the correct action depends on a quantifiable judgement (i.e. not nebulous "intuition"), and c) the decision can be made within the time available (i.e. the update cycle-time of the simulation).

The use of agents as active partners places several constraints on the design of the agent. To be effective as a partner for training humans, an agent should have the following qualities:

- **Correctness** - In order for the agent to relate the target strategy to the trainee, the agent must perform the strategy correctly.
- **Consistency** - The partner agent should also be consistent in its overall behavior. Inconsistency makes the learning process harder for the trainee.
- **Realism** - Relative to humans, agents have the potential to perform certain actions, make decisions, and respond with unnatural speed and accuracy, which is of less benefit for the purposes of demonstration. It is desirable for agents to exhibit more human-like behavior so that the performance appears achievable to trainees.
- **Exploration** - Exploration refers to how many different "situations" the agent gets the human into. Without exploration, the trainee cannot make a proper mental model or, for example, learn how to recover from errors simply because he has not experienced them.

How can a partner agent be designed to meet these desired criteria? There are a number of intelligent agent architectures that could be used to implement intelligent behaviors and decision-making within a simulated environment, including SOAR, PRS/dMARS, RETSINA, etc. Each architecture has a different approach to representing goals, domain knowledge, and actions. Each architecture defines a different mechanism for determining which sequence of actions it could take that would lead to accomplishing its goals by transforming the state of the world. Decision-making mechanisms range from reactive rule-based systems, to logical theorem-provers, to complex planning algorithms. In our approach, the appropriate knowledge is given to the agent for each sub-function of the task, representing cognitively distinct parts of the overall task (this would have to be based on a formal cognitive task analysis). Then, during training, the agent would perform one part of the task while the trainee performs the other.

Although agents can be used to satisfy the correctness and consistency criteria, something else must be added to introduce realism and exploration. *Our approach to adding realism to an agent's behavior is to artificially limit the speed and/or accuracy of responses by introducing stochastic errors*. For example, random delays could be added to response times, a small percentage of incorrect classifications could be added to a judgement/recognition task, or a small amount of imprecision could be added to a motor control component. The magnitude and frequency of mistakes made by the agent can be calibrated to measurements of human levels. These errors produce a variance in the agent's behavior that is important for both *realism* (humans often make mistakes, especially under high task-load) and *exploration* (allowing trainees to experience different parts of the problem space and practice recovering from errors).

Experiments

Space Fortress

To test our hypotheses, we implemented a partner agent for training human subjects in Space Fortress. Space Fortress is a laboratory task that was designed by researchers at the University of Illinois in the 1980's as representative of complex tasks for experiments with human performance and learning. The subject controls a "space ship" on a computer screen (see Figure 1). The ship may be rotated using a joystick, and it can move forward (in whatever direction it is pointing) by pressing forward on the joystick to fire a thruster. The ship can also fire "missles" by pressing a button on the joystick. There is a "fortress" in the center of the screen that cannot move, but can rotate and fire shells back at the ship to defend itself. The primary goal of the task is to destroy the fortress (as quickly as possible) without being destroyed. It takes ten single-shots (no faster than 250ms apart) followed by a double-shot (within 250ms) to destroy the fortress.

In addition to the fortress, another hazard in this environment consists of mines that appear randomly and float through the space, attracted toward the ship. When a mine appears, the subject must make a judgement about whether it is a friend or enemy mine before shooting at it (IFF: identify friend-or-foe). This is determined by ASCII characters that appear on screen. Prior to the task, the subject is given three characters to remember. When one of the letters in the memory-set appears on screen with a mine, the mine is a foe; for all other characters, the mine is a friend. The subject must double-click the right mouse-button within certain time constraints to indicate a foe, and then shoot the mine. If the subject makes the wrong choice (or does not respond quickly enough), the mine becomes indestructible and will continue to pursue

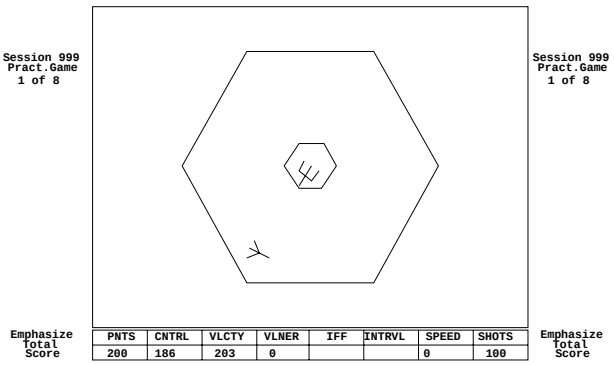

Figure 1: Space Fortress display.

the ship until it hits it (causing loss of points, and sometimes destruction of the ship) or times out and disappears.

Formally, the goal of the game is to maximize the Total score. The Total score is the sum of four sub-scores: Points, Velocity, Control, and Speed (indicated at the bottom of the screen for instant feedback). Points reflects the overt objectives of the game; the Points score increases for shooting and destroying the fortress and mines, and decreases for getting hit by or destroyed by them. Velocity scores are awarded for keeping the ship below a certain speed threshold. Control points are awarded for keeping the ship within the corridor between the two hexagons on screen (as opposed to flying a linear trajectory and wrapping around the screen). Speed reflects the timing of IFF judgements. In addition, subjects must maintain awareness of their supply of ammunition (missles). From time to time bonus opportunities appear (which the subjects must learn to recognize) and a decision must be made whether to add missiles to the ammunition supply or add points to the score.

Space Fortress represents a challenging task, both in terms of motor control as well as cognitive demands. Subjects must learn motor skills such as "tapping" the joystick to orient the ship (e.g. for accurate aiming), timed responses on button presses (e.g. for firing missles, IFF). In addition, Space Fortress has cognitive complexity involving memory, perception, attention, and decision-making. Examples include: a) following the rules of the game (different responses for friend vs. foe mines; 10 shots followed by a double shot to destroy the fortress, etc.), b) memorizing characters that indicate friend mines, c) keeping track of shots and missles, and d) deciding how to make bonus selections (for details, see (Mane and Donchin, 1989)). Furthermore, position, trajectory, and velocity are manipulated through acceleration (thrust) only, making it second-order control, which requires complex mental (spatial) calculations of vectors. Because of all these demands, Space Fortress typically takes on the order of 10 hours to learn. Novices typically score around -2000 in their first trials, and can reach scores as high as 5000-6000 (experts) asymptotically after training.

Implementation of the Partner Agent

We implemented an intelligent agent within the Space Fortress game that could control various parts of the game autonomously. The function-decomposition was based on a cognitive task analysis by Frederikson and White (Frederiksen and White, 1989), who analyzed the cognitive components of the overall task (e.g. navigation, aiming and firing, dealing with mines, managing missle resources). These tend to involve either the mouse or the joystick, without much interaction, so they can be easily separated. The agent was made able to control the direction and velocity of the ship and fire missles, as a human could do with the joystick, and the agent is able to manage IFF and select bonuses, as a human could do with the mouse. These functions can be decoupled and controlled independently, so the agent can act as a partner to the human by controlling one device-function while the human manipulates the other.

The agent was implemented by modifying the Space-Fortress source code (written in the C language) to mimic and over-ride inputs from the physical controls (joystick and mouse). The game is designed to run on a 46 ms update cycle, during which: a) inputs from the devices are sampled, b) state parameters (e.g. velocity, orientation) of objects are modified, c) the positions of objects on the screen are updated, and d) the game scores are revised. During an update, the actions available to the agent are: turn the ship, thrust, fire a missle, identify mines as friend or foe, or make a bonus selection. The agent implements a decision-making procedure that involves evaluating a number of conditions, such as speed, distance from fortress, appearance of mines, time since last button-press, number of missles left, etc., to determine which action is most appropriate to take at any given time.

The initial implementation of this decision-making procedure created an agent whose performance was so good that it did better than even the best-trained humans (scores around 8000, demonstrating a perfect strategy). However, we wanted to simulate a more natural level of expertise. The basic strategy for accomplishing this was to add a degree of randomness to the agent's simulated control inputs. For example, humans (especially novices) do not fire at the SF at exactly 250ms intervals. Furthermore, they do not always thrust in the ideal direction or for the perfect amount of time to properly control their trajectory. Thus, the decision procedure was modified by adding randomness to following aspects: *delay to identify mines* - random between 0.2s and 1.0s (uniform); *delay in firing* - follows a Poisson distribution with a mean of $\sim$1s; *variance in thrust applied* - range of 1-3 cycles, with mean of 2 (80ms); *precision of aiming* - rotations within $\pm 5^{\circ}$ of desired.

The behavior produced by this second version of the agent was more realistic and can generate scores at the level that the best humans (i.e. "experts") can achieve after training (around 5000 points, mean total score of top 5% of humans after 100 games by the standard training protocol). The agent appears to take essentially the

correct actions, with small but believable imperfections in navigation and firing; it takes a little longer for the agent to destroy the fortress, but it still usually destroys the fortress before making one complete cycle around the hexagon.

Design of the Initial Experiment

In this experiment, the hypothesis we were testing is: *Does training with a Partner Agent provide increased performance over training with other types of (individual) training?* For comparison, we evaluated the effects of training with the Partner Agent with a standard control training protocol, in which trainees simply practiced the whole task by themselves, trying to maximize Total score All trainees received the same standard instructions (written and on video tape) and two initial practice sessions for exposure, followed by an opportunity to ask questions. Then all trainees performed the task for 10 3-hour sessions spread over 4 days, where each session consisted of 8 three-minute trials, followed by 2 test trials, interspersed with rest breaks. Trainees following the Partner Agent protocol were randomly assigned to start with either the joystick or the mouse, and thereafter alternated roles with the agent on each trial. As subjects, 40 male students from the Department of Psychology at Wright State University were selected who played video games less than 20 hours per week; 20 subjects were assigned randomly to each protocol.

Results

The results of the experiment are shown in Figure 2. The graph shows the average scores for the two test trials at the end of each session (the baseline is the score for the practice trials before the first session; indicated as Session 0 in the graph). The scores are averaged over the 20 subjects in each group, with error bars indicating standard error for each measurement. It is clear that the Partner Agent protocol led to a higher asymptotic performance after 10 sessions of training that the standard control protocol. The final difference in performance between the two groups is significant at the $p < 0.05$ level by paired T-test ($t = 2.23 > 2.04$, $df = 38$).

When performance is decomposed into sub-scores (Table 1), trainees with the Partner Agent were found to do better on each component after 10 sessions of training than those who used the standard control protocol, although the differences were only statistically significant for the Velocity and Speed scores (lack of significance for Control and Points scores is potentially due to the high variance and small sample sizes). The reason that the magnitude of improvement was greater for some sub-scores than others may reflect differential impacts of observation versus hands-on practice for sub-skills, depending on the degree to which they rely on implicit or explicit processing during the approach to automaticity (Goettl et al., 1997).

Effect of Level of Expertise

The success of the first experiment led to a follow-up question: *Does the level of expertise simulated by the Partner Agent affect the magnitude of performance improvement with training?* We hypothesized that it would, based on evidence that trainees model the behavior of their partners, and the performance improvement is correlated with the intelligence of their partner. To test this hypothesis, we created three variants of the agent, simulating three different levels of expertise: novice, intermediate, and expert. These behaviors were defined operationally in terms of the following reference groups: "experts" were defined as those who achieved scores in the top 5% after 10 sessions of training, "novices" were the baseline scores of these subjects prior to training (equivalent to the whole pool of trainees, since those who became experts were not distinguishable), and "intermediates" were defined as trainees who had reached a Total score halfway between novice (baseline) and expert (which could occur in any session). Novices tend to lack control, fly too fast, go outside the hexagon, and even wrap around, missing the fortress and mines with shots, and getting destroyed more often than destroying the fortress, whereas experts rarely get hit, navigate slowly around the fortress with many incremental thrusts, and typically destroy the fortress within one complete pass.

The different behaviors of the Partner Agent were generated by adding different amounts of randomness to the timing and precision of the agent's actions. For example, the time delay in identifying mines was varied from around 0.5s (for experts) up to around 4s (for novices; just at the boundary of when mines time-out). Aiming accuracy ranged between $\pm 5^\circ$ (experts) and $\pm 8^\circ$ (novices). Thrust durations ranged from a mean of 80ms

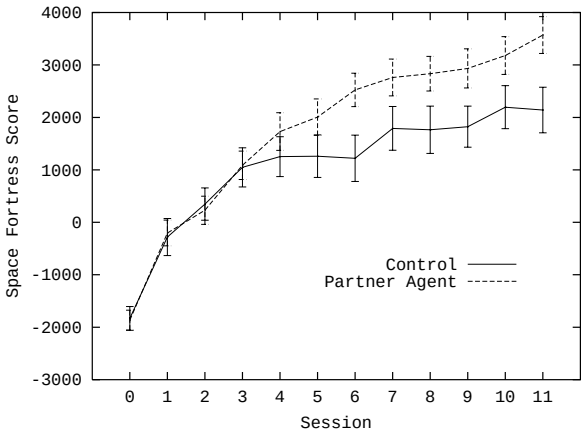

Figure 2: Results from the first experiment.

Table 1: Final sub-scores after 10 session of training (with standard errors in parentheses).

	Control Protocol	Partner Agent
Velocity	49.7 (215.7)	709.1 (136.6)
Control	903.2 (94.8)	1077.9 (50.1)
Speed	438.3 (69.7)	639.5 (49.6)
Points	750.2 (174.4)	1143.5 (189.8)
Total	2141.4 (435.0)	3570.0 (351.3)

Table 2: Target skill levels for defining behaviors of Novice, Intermediate, and Expert agents.

agent	Points	Control	Velocity	Speed	Total
Novice	-863	462	-388	21	-768
Inter.	412	1072	673	490	2645
Expert	2314	1229	1132	958	5633

Table 3: Results of second experiment, showing Total scores after training (10 sessions) with Partner Agents of different skill levels.

Protocol	Mean	Std. Err.	Participants
Expert Agt.	3611	514	15
Intermed. Agt.	2306	475	18
Novice Agt.	2120	587	13
Control	2034	467	19

(experts; shorter impulses create finer control) to 320ms (novices). And the delay for bonus selection ranged from around 0.4s (expert) to around 9s (novices), with intermediates making judgements at around 4s. No errors were introduced into the correctness of IFF judgements, as humans are rarely observed to make mistakes at this. The behavior of each agent was calibrated against the scores and sub-scores of the target group (shown in Table 2) by adjusting the internal parameters until the scores were within one standard deviation (see (Sims, 2002)).

This second experiment was conducted much like the first. Male psychology students were drawn as subjects and randomly assigned to one of four groups: control (standard instruction and practice), novice partner agent, intermediate partner agent, and expert partner agent. Trainees in each group performed 10 sessions of training, each consisting of 8 three-minute trials followed by two test trials. Table 3 shows the final scores, after the 10 sessions, averaged over all subjects in each group. The results indicate that training with the expert Partner Agent provided the most benefit, and that simulating intermediate or novice levels of expertise did not provide improvements over the standard control protocol.

Discussion

The fact that training with a Partner Agent improves the asymptotic performance of trainees over those who use the standard control protocol contrasts with earlier experiments with the AIM-Dyad protocol (with human partners). AIM-Dyad has consistently been shown to reach *the same* levels of performance as the control protocol (Arthur et al., 1997). The advantage of it is that partner-based training requires fewer resources, such as instructors, workstations, etc. Thus, it can be said the AIM-Dyad provides an improvement in training *efficiency* but not *effectiveness*. While the Partner Agent protocol looses the gain in efficiency (since humans are training individually again), it demonstrates an increase in training *effectiveness*, based on the statistically significantly higher scores reached by trainees after 10 sessions.

The methods for developing the Partner Agent used in these experiments could potentially be used for building intelligent training systems for complex tasks in other domains too. The primary requirement is the availability of a cognitive task analysis to define the subcomponents of the task (at a cognitive level). These functions would then be automated through agent programming, e.g. by building a knowledge base of goals and actions for achieving them that could be used for making decisions. The behavior of the agent can be made more realistic (human-like) by introducing artificial errors (inaccuracy) and random time delays in actions taken by the agent. While this was accomplished in this work by manually tuning internal parameters for randomness to match the performance of the agent to target human groups, it might also be possible to use more automated methods to capture human-like strategies and behaviors (including errors) directly from transcripts of human-performance data, such as by using reinforcement learning (Kaelbling et al., 1996).

References

Anderson, J., Boyle, C., Corbett, A., and Lewis, M. (1990). Cognitive modelling and intelligent tutoring. *Artificial Intelligence*, 42:7–49.

Arthur, W., Day, E., Bennett, W., McNelly, T., and Jordan, J. (1997). Dyadic versus individual training protocols: Loss and reacquisition of a complex skill. *Jour. of Applied Psychology*, 82(5):783–791.

Bandura, A. (1986). *Social Foundations of Thought and Action: A Social Cognitive Theory*. Prentice Hall, Englewood Cliffs, NJ.

Frederiksen, J. and White, B. (1989). An approach to training based upon principled task decomposition. *Acta Psychologica*, 71:89–146.

Goettl, B., Snooks, S., Day, E., and Shebilske, W. (1997). Emphasis change and verbal elaboration in skill acquisition: A tale of two components. In *Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting*, pages 1195–1199.

Kaelbling, L., Littman, M., and Moore, A. (1996). Reinforcement learning: A survey. *Artificial Intelligence Research*, 4:237–285.

Mane, A. and Donchin, E. (1989). The Space Fortress game. *Acta Psychologica*, 71:17–22.

Rickel, J. and Johnson, W. (1999). Virtual humans for team training in virtual reality. In *Proceedings of Ninth World Conference on AI in Education*, pages 578–585.

Schneider, W. (1985). Training high-performance skills: Fallacies and guidelines. *Human Factors*, 27(3):285–300.

Shebilske, W., Jordan, J., Goettl, B., and Day, E. (1999). Cognitive and social influences in training teams for complex skills. *Journal of Experimental Psychology: Applied*, 5:227–249.

Shebilske, W., Regian, J., W. Arthur, J., and Jordan, J. (1992). A dyadic protocol for training complex skills. *Human Factors*, 34(3):369–374.

Sims, J. (2002). The use of partner agents in training systems for complex tasks. MS thesis, Department of Computer Science, Texas A&M University.

Wooldridge, M. and Jennings, N. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152.