Gary Stein and Avelino J. Gonzalez*

Building and Improving Tactical Agents in Real Time through a Haptic-Based Interface

Abstract: This article describes and evaluates an approach to create and/or improve tactical agents through direct human interaction in real time through a force-feedback haptic device. This concept takes advantage of a force-feedback joystick to enhance motor skill and decision-making transfer from the human to the agent in real time. Haptic devices have been shown to have high bandwidth and sensitivity. Experiments are described for this new approach, named Instructional Learning. It is used both as a way to build agents from scratch as well as to improve and/or correct agents built through other means. The approach is evaluated through experiments that involve three applications of increasing complexity – chasing a fleer (Chaser), shepherding a flock of sheep into a pen (Sheep), and driving a virtual automobile (Car) through a simulated road network. The results indicate that in some instances, instructional learning can successfully create agents under some circumstances. However, instructional learning failed to build and/or improve agents in other instances. The Instructional Learning approach, the experiments, and their results are described and extensively discussed.

Keywords: Haptics, instructional learning, multimodal machine learning, tactical agents.

DOI 10.1515/jisys-2014-0126
Received August 28, 2014; previously published online March 10, 2015.

1 Introduction and Background of our Research

The automated creation of intelligent and autonomous agents for use in tactical simulations and robotic applications has long been a goal of computer science research. By intelligent agents in this context, we mean programs that can perform human-like control tasks at the lowest level (motor skills), as well as make high-level decisions on how and when to use those motor skills when performing a complex task or conducting a mission. We refer to these as *tactical agents*.

Machine-learning techniques have been used by many to make tactical agents learn while they execute in an environment, typically military simulations, computer games, or RoboCup soccer. Reinforcement learning (RL) has been the preferred means of training these agents ([2, 3, 15, 27, 33–35, 45, 54, 56, 57], among many others), although some work in evolutionary computation and neural networks has also been evident in the literature [41, 48]. We are interested here in using machine-learning techniques with haptic devices to automatically (or semi-automatically) create and improve agents that perform tactical tasks competently.

Much work has also been done to create tactical agents through learning from observation/learning by demonstration ([6, 8–10, 14, 18, 26, 29, 37–39, 42, 44, 55], among many others). In this approach, the agent learns a task by observing a human performing the task or mission, preferably without any interaction other than the observation itself. However, some problems persist with these techniques for teaching agents how to behave tactically. First, it can be difficult for these agents to generalize their actions when facing situations that have not been explicitly included in the training set. This can lead to incorrect behaviors when the

***Corresponding author: Avelino J. Gonzalez,** Intelligent Systems Laboratory, University of Central Florida, 4000 Central Florida Boulevard, Orlando, FL 32816-2362, USA, e-mail: Avelino.gonzalez@ucf.edu
Gary Stein: Primal Innovation, 201 Tech Drive, Sanford, FL 32771, USA

resulting agent is faced with a previously unseen situation. While generalization of the learned behavior has been achieved to a degree [23, for example], it depends on multiple demonstrations of related actions done in a slightly different manner. Second, it can often be difficult to "repair" an agent after its creation. Typically, if an error or a gap in the agent's behavior is identified, the common solution has been to introduce more training samples/demonstrations and rerun the system. These errors can often be the result of faulty human performance. To address this issue, Grollman and Billard [21] reported a method that can use failed demonstrations to teach the robot what not to do. While this may possibly be a way to repair errors or fill in gaps in knowledge, in general, retraining can involve significant effort and become expensive to do. Lastly, validation of the produced agent can be challenging because such agents perform actions continuously. While validation approaches for continuous systems do exist, they depend on comparison with human performance traces or inspection by humans in real time.

We seek to address the first two of the above problems through real-time instruction, where a human instructor observes the actions of an existing tactical agent in a simulation, and employs a haptic force-feedback device to provide corrective counterforce when he/she deems it necessary. This is done in real time as the instructor watches the agent perform the actions. This enhanced training can be used to create an agent from scratch and/or to teach the agent how to behave in situations not seen in the training set, thereby enhancing its generalization. In the same manner, an agent's flawed actions could be corrected and/or upgraded through such sessions with instructors.

Our hypothesis, therefore, is that real-time instruction via a haptic device can be used to (i) successfully build agents from scratch that perform certain tasks proficiently, and (ii) improve agents already built through other means. We refer to this learning approach as Instructional Learning (IL).

IL has been discussed in the literature [11, 19, 20, 30, 43] as a way to create intelligent agents without any direct programming. However, there is no generally accepted definition of this term. As in RL, our approach trains an agent to perform a task by penalizing the agent when its actions differ from what the instructor would do under the observed conditions. The difference is that the penalty is made known when the instructor, upon seeing incorrect behavior by the agent, takes over the control of the agent by pressing and holding down a button on the haptic device, and applies a corrective counterforce to the agent to appropriately control it. No counterforce is applied by the instructor when he/she approves of the actions of the agent. The amount of penalty is directly proportional to the intensity of the counterforce exerted, and this penalty is used to modify the agent's behavior to make it act correctly.

In summary, we believe that IL is an underdeveloped training approach that could be useful for creating tactical agents. We proceed to evaluate our hypothesis above by building a prototype that implements our IL approach through a haptic interface (a force-feedback joystick). We assess the effectiveness of our approach quantitatively and qualitatively, and report the results in this article.

We next discuss haptics and the kinds of devices that can be useful in our research. Section 3 describes the testbed applications used to evaluate our work, followed by an in-depth description in Section 4 of our approach to IL and the prototype IL system (ILS) that was built to evaluate our ideas. Finally, the remaining chapters discuss the results of the tests performed, followed by a concluding chapter.

2 A Brief Discussion of Haptics

To gain an understanding of how our IL process works, some background in haptics is essential. In this section, we briefly discuss the state of the art in haptic interfaces and their capabilities.

2.1 Haptics

Haptics is the study of the sense of touch. In computer terms, haptics refers to interaction with a computer system through touch. Srinivasan and Basdogan [47] define a haptic interface as one that "displays tactual

sensory information to the user by appropriately stimulating his or her tactile and kinesthetic sensory systems." Force-feedback systems are a class of haptic devices that, as the name implies, provide feedback on their movements. These devices can convey a force through embedded motors to stimulate the sense of feel. An example of force-feedback device is a car steering wheel in a simulation that provides artificial, yet realistic, resistance when turning. Haptics are used in simulation and in virtual reality to provide a better sense of realism. These devices provide additional sensory cues that are presented to a human in a natural way. The extra information obtained through haptics can be combined with computer visuals to help in the decision-making process [36].

Research in haptic interaction with a computer has been ongoing for over 50 years. Fisher et al. [17] state that "Haptic feedback is best for interfaces that are complex, sensorially overloaded, and require continuous control." Haanpaa and Boston [25] describe haptics as having a "high-bandwidth modality" and can "reduce mental workload."

Avizzano et al. [4] found that without error information, people have the tendency to repeat the same incorrect actions. However, when a haptic spring-based attractor was applied to correct the participants, they were able to improve their skill beyond what was achieved by having just the visual information. Feygin et al. [16] found that complex motor skills that are difficult to explain and describe verbally, or even visually, could be conveyed easily by haptic means. They also found that haptic guidance was better with respect to timing than visual methods.

2.2 Haptic Devices as Training Mechanisms

Haptics have been used for training in human-to-human, machine-to-human, and human-to-machine modes. Although our objective is human-to-machine skill transfer, insights can be found in mechanisms that have been applied to teaching in other ways. We describe some of this research below.

Human-to-human instruction is the traditional method of training, where haptics devices are used as the medium. The majority of these systems seem to be applied to medical training for surgery [7, 24]. On the other end of the spectrum, haptics has been used for rehabilitation of stroke victims. Dinse et al. [12] found that perceptual skills improve with practice for impaired individuals by systematic stimulation of a pattern through haptics.

Machine-to-human training reflects only a slight modification of human-to-human training. The hand is still directed through the haptic device, except that now no longer by a human teacher but by a virtual teacher. Gillespie et al. [19] attempt to train participants by having them feel an already-known optimal strategy and then learn that strategy themselves. They found that practice time could be reduced when a learner is exposed to the correct way of performing the control task at an early stage of the learning process. Therefore, if the learners feel and see an already known answer, they do not need to discover one for themselves.

Patton and Mussa-Ivaldi [40] use a force field generated by a computer through a haptic device of sorts to teach movements to humans who have suffered brain injury. In a similar fashion, Lee and Choi [32] employed a concept called haptic disturbance to facilitate this teaching.

Bayart et al. [5] found that full guidance outperformed no guidance with respect to position and improved timing accuracy. However, they also found that partial guidance outperformed full guidance. They attribute this to the idea that people learn from their mistakes. If the instructor assumes full control, no mistakes are made, and the student is unprepared when previously unseen issues arise.

Human-to-machine training has been investigated significantly less than the other two modes. Aleotti et al. [1] used haptics as part of a Programming by Demonstration approach, where vibration indicated to the instructor what the physical system could not do. Eguchi et al. [13] used haptics to show the instructor the differences between the virtual system and the actual robot to be trained. Again, this improved the teaching to create better trajectories that could be then learned through backpropagation. Goodrich and Quigley's system [20] learned to steer a car in order to stay within a lane. The system first started with a proportional-derivative (PD) controller to keep the car in the lane. Then, the instructor operated the vehicle while the PD

was running and the haptics of the system was based on the admittance control policy. Then, a Q-Learning reinforcement system was used to learn the corrective force the human used against the PD controller. In essence, the system learned when not to trust the control law and used the amount of force the instructor was willing to exert to overcome the PD control. This system learned from the human when to overrule the PD control system, but not specifically how to control the vehicle.

Calinon and Billard [8] reported success with a system that uses a teacher's physical guidance of the robot arm to help it learn a new motion. While this does not exactly employ a haptic device, it could be argued that the teacher uses a physical force to instruct the robot. However, this task is completely demonstrational and does not involve feedback. That is, there is no counterforce applied to the motion of a robot attempting to perform the intended task.

Lastly, Kucukyilmaz et al. [31] investigated the use of haptic devices to serve as communication of intent between a human and a robot cooperating on a task. Their work is designed to allow the robot to infer the intentions of its human collaborator, and adjust control levels accordingly, subject to negotiations between them, all done through the haptic device.

2.3 Our Approach to Using Haptics for Training Agents

None of the systems mentioned above uses the degree of corrective counterforce by an instructor as a cost function to train an agent to perform a task. This is what we do in our work described here, and it is what makes our approach unique. Note that we deal with two different haptic forces here: (i) the motion of the joystick felt by the instructor (This motion is produced on the joystick by the agent's actions, as if the agent was directly manipulating the joystick.); and (ii) the counterforce applied by the instructor when he/she takes over the control of the agent to correct the errant actions of the agent.

3 Experimental Testbed Applications

It may seem rather odd to discuss the evaluation testbeds before describing our ILS itself and its prototype. However, these testbeds are critical to our explanation, so we describe them here, albeit briefly, in the expectation that knowing about these will enhance the clarity of our description of IL, which is found in Section 4 below.

We evaluated the ILS prototype system in three different testbed applications – Chaser, Sheep, and Car. These testbed applications were first described in our prior paper [51], in which we report a different research that used the same testbed applications. These testbed applications involve teaching the agents about two-dimensional motion constrained by the laws of physics. The agents to be created involve performing motor skills, plus planning and tactical decision making. The physics for each simulation was calculated at 100 Hz to ensure close to continuous operation, even though it is, in fact, a discrete system.

3.1 Chaser

Chaser involves a basic task where the agent controls a chasing entity with a limited turning rate that attempts to capture a preprogrammed fleeing entity. Figure 1 shows what the instructor sees while the agent performs its actions. The larger square is the chasing entity controlled by the instructor, while the smaller square is the scripted fleeing entity. The small line emanating from the chaser indicates its instantaneous direction of motion. Linear and angular velocity are the only two variables controlled by the instructor through the joystick. The scripted fleeing entity follows these simple rules: (i) always attempt to move on a vector away from the chaser; (ii) always move faster than the chaser; and (iii) seek to stay near the center of the playing area. The fleeing entity has a small element of random movement that increases the difficulty of the problem. Two environmental inputs are presented to the agent: (i) the relative distance to the fleer and (ii) the relative angle difference with the fleer. Chaser is considered the simplest of the three domains, requiring only reactive skills.

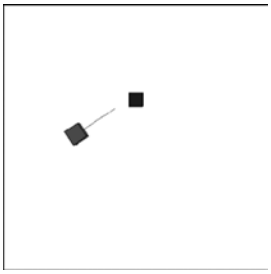

Figure 1: Chaser Interface (Originally Found in Reference [51]).

3.2 Sheep

The Sheep testbed application involves a shepherd agent trying to "shoo" its sheep into a circular pen at the center of the field. Once a sheep enters the pen, it cannot escape. The sheep, on the other hand, tend to run away from the shepherd agent, so planning on the shepherd agent's part is essential. In this application, the agent must herd 16 sheep into the pen. It is meant to test multiple objectives such as planning, as tracking and herding the sheep is best done on the herd, and not on each individual sheep. There are two control output variables: the linear and angular velocities. A circular fence acts as an outer barrier at the border of the playing field to prevent the sheep from running off the field.

The shepherd agent is presented with 11 environmental inputs. The first three inputs are for spatial awareness and include the distance from the center of the pen, the relative angle to the center, and the current angle of the agent with respect to north. The relative distance and angle help with the approach to the pen, while the actual angle could be used for setting up specific position-based maneuvers. The other eight inputs are based on the position of the sheep divided into pie wedges. Pie wedges are commonly used when dealing with multiple items in a scene without having the number of inputs change with the number of items. The space is divided into eight equal slices of 45° each, and centered at the shepherd agent. These wedges are relative and move based on the location and heading of the shepherd agent. See Figure 2 for a depiction of what the instructor would see (except for the pie wedges). The small inner circle is the pen, while the larger outer circle is the fence. The dark squares are the loose sheep, while the gray ones are those sheep already captured in the pen.

3.3 Car

Car is the most complex of the three testbed applications. It consists of controlling a car agent on a two-lane loop road with traffic, where the agent must keep the virtual car in the proper lane and at a proper distance from other cars, as well as make left turns at two intersections. This task involves dynamic obstacles and more

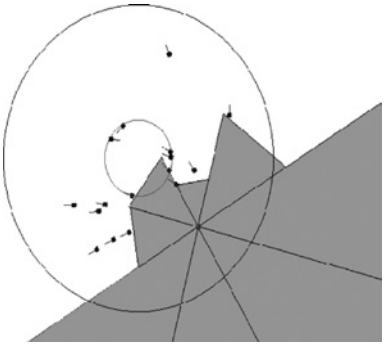

Figure 2: Sheep Interface (Originally Found in Reference [52]).

complex physics, as well as more salient decision making (when to commit to making a left turn in the face of oncoming traffic).

The agent receives 11 inputs from the environment. The first two inputs are based on the X and Y locations of the vehicle on the track. This information can be used to differentiate turning operations or to regulate the speed of the car at different locations on the track. The next input is the relative angle of a "carrot point" down the road. A carrot point is located on the center of the lane at a location a fixed distance in front of the car. This point can be used for steering to stay in a lane. The other eight inputs are from the pie wedges around the car. Each wedge computes the distance between the agent and any other car within its discrete sensor range, both in the merging and normal driving situations. The wedges include blind spots when a car is behind another car and not directly visible. Figure 3 shows the interface seen by the instructor. The clear box is the car driven by the agent, while the dark boxes represent other traffic that is scripted to simply go around the track along the lanes provided.

3.4 Assessment of Agent Performance

The errors that an instructor is to correct are those of actions leading to improper time and space. These differ in the context of each of the domains; however, for example, in the case of Chaser, obvious errors would include the chaser moving in the opposite direction of the fleer's motion. In Car, it could mean the agent's car turning left and colliding with an on-coming vehicle. Such errors would reflect a lack of motor skills, inappropriate decision making, or both. The performance of an agent would be reflected in how well it performed the tasks and avoided such errors.

To analyze the performance of the resulting agents, we created the performance rating (PR) as a metric through which we could compare the performance of agents built through various learning approaches, including IL. The PR is defined as a domain-specific formula used to calculate how successfully a task is being accomplished by the agent (or even by a human actor). The PR enables one to objectively measure the performance of the created agent. Nevertheless, an informal definition of how agent performance was assessed follows. A full discussion of PR for these three domains can be found in Reference [50]. Note that in Stein [50], PR is referred to as "Fitness." We changed the label in Reference [51] and in this paper to more descriptively reflect what this metric is.

In the Chaser simulation, the PR of the agent is based on the physical distance between chaser and fleer, measured at every time step, with a score of 1.0 if the two coincide and 0.0 if they are the maximum distance apart. The PR is the sum of all the time step scores scaled to the duration of the event in seconds (30 s). In the Sheep simulation, the performance is based on the number of sheep that entered the pen, with partial credit for getting the sheep near the pen. This PR is also scaled to the duration of the simulation (60 s in this case). The Car agent's performance is graded on several factors that include positive points for the number of laps completed while in the proper lane, and negative points for time spent tailgating the car in front, forcing traffic behind to slow down, turning left too close to on-coming cars, and of course, for collisions. The negative points for tailgating, slowing traffic, and causing on-coming cars to brake were limited such that the agent could lose at most one lap; a collision, however, causes the agent to lose all laps. The distance traveled

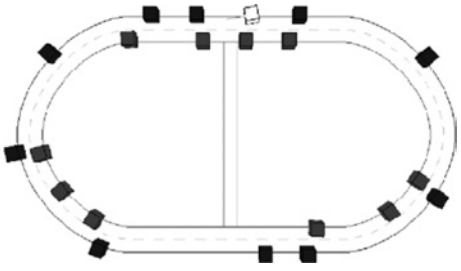

Figure 3: Car Simulation Interface (Originally Found in Reference [51]).

was likewise scaled to event duration time (also 60 s). The maximum number of laps achievable was five, based on the speed of the car and the 60-s duration of the exercise.

4 IL and its Implementation

Here, we describe our IL approach and how it was implemented in our prototype system. We begin by briefly discussing the work of others in IL, and follow this with some general concepts before describing the haptics force-feedback algorithm and the machine-learning algorithm employed.

4.1 Work of Others in IL

There are several ways to present the agent-instructor relationship for a motor skill learning task. Gillespie et al. [19] describe three similar methods that use the example of someone learning to play tennis. Their (three) different instructional paradigms indicate different approaches for how to teach the game. It can be interpreted from their work that the demonstration of the motion of the racket is important, but so is feeling the force while executing that motion. Additionally, the instructor's hand can provide live corrections of the student's motion, where the student can "feel" the feedback. Gillespie et al. [19] implement the motion and feel through a force-feedback device. It is important to note, however, that in their system, a computer is training a human, rather than the reverse, which is our objective here. Nevertheless, their concept served as inspiration for our work in humans training computer agents.

Schaal [43] implements an instructional system on a robot platform. In his system, the author attempts to overcome the old method of observation that "normally consisted of manually pushing the robot through a movement sequence." [43] They attempt to learn a control policy that is "bootstrapped by watching a demonstration" [43], but then follows the teachers' trajectories by imitation while adjusting for kinematics and dynamics. This is different from our work in many ways, as we do not use observation and our instruction is done through haptics on tactical agents. Kosuge et al. [30] use a hybrid method that combines force and position control. Using the force applied, the system creates a set of control laws that monitor the input force in addition to the position.

We believe that in some applications, it is possible to build a competent agent from scratch strictly through IL, starting with a highly erratic (possibly random) agent initially. On the other hand, as Schaal showed, IL need not exist as a stand-alone approach to training agents [43]. If a tactical agent could already perform a task and only needs to be enhanced and/or improved, the IL process becomes more practical because it corrects only those behaviors that are incorrect.

As an on-line training strategy, RL matches up well with our IL and the two are synergistic. RL algorithms work by using positive reinforcement to actively encourage certain policies in an agent. IL provides an error function to drive the RL mechanism.

One problem with IL is that it requires a person to actively instruct the agent in real time. This means that, for example, if 30 hours of training time were to be needed for some agent training application, an instructor could be tied up for several days. Therefore, we believe that IL is best used as part of a two-stage agent building process that uses other means to initially bootstrap the agent's behavior to a certain level of competence, and then IL is performed to enhance and/or improve its performance. This can serve to reduce the amount of instruction time by pruning out poor-performing and/or untrained initial agents.

4.2 General Concepts

The basic idea behind our interpretation of IL is that a human instructor observes, through some interface such as in Figures 1–3, the actions of the agent being created. In addition, the agent's actions are also reflected

as motion and force in the force-feedback haptic device, just as a human would physically move the joystick if he/she were controlling the agent.

The instructor acts as an evaluator of the agent's performance and expresses his/her assessment as corrective feedback. Note that he/she does not override the actions of the agent – only adds or subtracts from it until the agent behaves as the instructor desires. This feedback is communicated as counterforce to the actions of the agent via the haptics algorithm. The greater the counterforce is, the more correction that is deemed necessary by the instructor. In contrast, no counterforce at all implies no correction is necessary.

The agent is modified in real time by a machine-learning algorithm that changes the agent's code to reflect the feedback just provided by the instructor. Note that the machine-learning algorithm is compatible with the agents' structure so that it is able to modify the structure of the agent to achieve the desired results. The difficulty was in how to implement these general ideas into a system. We describe this next.

4.3 Haptics Algorithm

A library was written to independently send the agent's operation as joystick motion to each joystick axis at the same time, given the desired position. A control loop physically implements the actions of the agent, which can then be felt by the human instructor on the force-feedback joystick. This agent action is thusly updated to the joystick motors at 100 Hz. These motors then implement the X-Y position on the joystick that would be required in order to achieve the desired action. Because the motors have low torque, the instructor can physically (and rather easily) overpower the joystick. However, if the human was not holding the joystick, one could actually see the joystick moving on its own, in real time and in accordance with the actions seen on the screen. The force at the center of the joystick handle is approximately 2 kg_F, and the forces act at the update rate of 100 Hz. However, the physical position of the handle might lag the agent by approximately 0.25 s because of the low torque involved.

Our system uses an evolutionary approach to teach the agents (see the description of our machine-learning algorithm called PIGEON below). This means that several individual agents will be vying with each other to be the best one at every step of the way. The agents operate like "thirty monkeys attempting to learn to drive a bus" [46], all vying to be the controller. Therefore, the force feedback felt by the human instructor in the joystick is the output action of the current "champion" agent. Thus, the joystick would attempt to move to the X-Y position of the champion agent with a force vector from the current X-Y position. The human instructor has the option of either allowing the current champion to "drive," or to take over the control of the agent by holding down the trigger button in the joystick and moving the joystick in the proper direction. The instructor, therefore, needs to exert more force than the agent and hold the joystick at the new desired position, while continuing to hold down the trigger button.

While the instructor can show the agent how to get out of tricky situations, he/she could deem the training session a failure and revert the agent structure back to what it was at the beginning of the session. We arbitrarily gave the instructor 20 usable real-time training sessions of an agent in which the instructor can give and receive feedback. However, he/she could have more sessions if he/she deemed necessary and so requested.

The description of this process is given in pseudo code in Algorithm 1 and is further described in Section 4.4 below.

4.4 The PIGEON Machine-Learning Algorithm

The machine-learning algorithm used is called PIGEON (stands for Particle swarm Intelligence and Genetic algorithms for the Evolution and Optimization of Neural networks). It takes the difference in performance noted by the system and modifies the agent structure to reflect the instructor's feedback. While a description of the PIGEON learning algorithm is essential to our discussion, it is not the emphasis of this paper. We refer

Algorithm 1 – IL Algorithm

```
Ensure: Population didactic reward of all individuals is 0.0
for t=0 : Dt : Time do
  HighestReward=0.0
  if TriggerPressed then
    HumanControl=1
  else
    HumanControl=0
  end if
  for j=0 : PopulationSize–1 do
    ComputeAction(Agent_j)
 if HumanControl=1 then
    CurrReward=HumanReward ()*Dt
 else
    CurrReward=Dt
 end if
 Fitness_j=Fitness_j+CurrReward
 if CurrReward>HighestReward then
    HighestReward=CurrReward
    BestAgent=Agentj
 end if
 end for
if RewardAtEnd=0 and OneSecondInterval(t) then
 EvaluatePIGEON() //Algorithm #2; see Section 5.3 for description of PIGEON
end if
if HumanControl=0 then
 DoActions(BestAgent)
else
 DoActions(Human)
end if
ForceFeedback(BestAgent)
ComputeSimulation(Dt)
end for
 if RewardAtEnd=1 then
EvaluatePIGEON()
 end if
```

the reader to References [50] and [52] for full details of PIGEON. Nevertheless, for the sake of completeness, a brief introduction of the PIGEON algorithm follows.

The PIGEON algorithm (just PIGEON for short) is composed of two other well-known machine-learning algorithms, NeuroEvolution of Augmenting Topologies (NEAT) [49] and Particle Swarm Optimization (PSO) [28]. NEAT is an artificial neural network (ANN) training system that uses genetic algorithms (GAs) to modify the structure and evolve a population of individual ANNs. PIGEON uses PSO intermittently to optimize the weights of the ANNs built by NEAT. The high-level PIGEON algorithm is shown below as Algorithm 2.

The NEAT algorithm part of PIGEON creates and trains arbitrarily connected neural networks. It uses a set of generic markers called *innovation numbers* to evolve the structure of the ANN. This is done by adding new links or nodes in a coherent way that allows for *mating* to happen later. NEAT follows a generational GA model that combines well-performing individuals to produce the next generation. This population of individuals evolves through mating, where offspring are produced through swapping the weights tagged with the same innovation numbers of the two parents, and random mutation that creates new structures. Those individuals with higher performance (*fitness*) have a higher probability of mating.

PSO operates more as a system of social forces that manipulates a population of individuals. The algorithm treats each individual and its set of weights as a particle with a position in the solution space. Similar to GAs, PSO requires a fitness function to evaluate performance. The system then sets up a series of forces in the direction of the global best individual and toward the best past configuration of an individual. Because

Algorithm 2 PIGEON algorithm (originally found in Reference [52])
1: **for** Generation=0: TotalGenerations/HandOff **do**
2: **for** Iteration=0: HandOff–1 **do**
3: // HandOff is the time when NEAT hands off control to PSO
4: PSO Iteration()
5: **end for**
6: NEAT Generation()
7: **end for**

the particles are given a force in a second-order system, integration must be done to calculate their velocity and their next position in an iterative fashion. On each iteration, the fitness of all the individuals is calculated and the particles are given the appropriate force.

PIGEON extends the NEAT algorithm by using the same genetic markers to line up and perform this PSO update iteration. The genes that make up the "DNA" of the individual are grouped together based on the innovation numbers. Then, there is a combination of the current individual, the individual's best past configuration, and the best individual of the group. The weight updates follow the traditional PSO mechanism; however, this process allows heterogeneous populations to be potentially evaluated. As part of the PIGEON process, the PSO evaluation happens more often than the NEAT structural changes. Algorithm 2 shows how PIGEON works at a very high level. For every generation of the NEAT algorithm, a number of PSO update iterations are applied. This number is determined by the variable HandOff. Several values of HandOff were tested, but the best value of HandOff was found to be 5.

Our IL approach uses a population of agents that are graded together by the instructor. The collection of agents eventually learns the task by maximizing the rewards from the instructor. PIGEON is particularly relevant because it is based on PSO and its ability to learn quickly. This feature is important because the haptic interaction with the trainer happens in real-time. During IL, the actions of all of the agents of the population are evaluated and sorted based on similarity to the human instructor over a set synchronization time of 1 s. The agent that best matches the human action is selected as the "champion agent" whose actions are reflected in the haptic device. The weights of the ANN that makes up the champion agent are known during that 1-s period. The weights of all individual ANNs are then adjusted using PSO. The PSO technique was selected for its fast convergence properties and its quick computation time [22]. Both of these factors are important for real-time learning because the heavy generational computation of other algorithms, such as NEAT, do not lend themselves well to this type of real-time operation. Additionally, real-time techniques such as rtNEAT [48] are not applicable because of the synchronous nature of the problem. The agents created here experience real-time adaptation during training.

4.5 Testbed Hardware Used

A force-feedback joystick was used as the haptic interface between instructor and computer. The joystick selected was the Logitech Force 3D Pro (Figure 4) as a commercial off-the-shelf device. This joystick provides geared motors on the X- and Y-axes that can implement forces independently and are not merely adjustable spring tensioning to center. These motors can be driven at 100 Hz from the computer. When implementing the above algorithm, the instructor uses the front trigger button to tell the system if he/she wants to provide corrective feedback. The joysticks provide 2 kg_F of force, which is not enough force to overpower the human instructor at any time. The force-feedback strategy uses a directed "spring constant" vector from the position of the joystick as held by the human trainer to the top agent as sorted by the Didactic Reward method (explained in Section 4.6 below).

The entire system was implemented on a desktop computer running Linux using a three-dimensional interface written in OpenGL. This provided the user with a visual environment using the third-person perspective that could be easily understood by those who play video games.

Figure 4: Logitech Force 3D Pro.

4.6 Reward Assignment Comparison

As part of the IL approach, the agents are continually being given learning rewards and penalties by the instructor. This is termed the Didactic Reward and is based on the amount of counterforce that the instructor exerts to correct the agent at each time step, scaled linearly from 0.0 (no force) to 1.0 (maximum force), multiplied by the time step (Dt).

From the above, full reward is given when no counterforce is exerted by the instructor. This is represented by lines 10 to 14 in Algorithm 1 above. The agents are sorted based on these rewards, and the best-rated agent is placed at the top of the list.

However, there are competing ideas for how best to integrate these rewards into our IL process. Should all the reward be given at the end of a training session in a batch fashion (Reward-at-End)? Or should the reward be distributed during the run (Reward-During)? Regardless of the method of giving rewards, the agents are trained using the PIGEON algorithm with a fixed network during the run and weights adjusted using PSO. The two algorithms are related by implementing one loop of Algorithm 1 every time the Evaluate PIGEON function is called (note lines 22 and 33). Additionally, Algorithm 1 is performed for both types of IL with RewardAtEnd equaling 0 or 1 for Reward-During or Reward-at-End, respectively (note lines 21 and 32).

The two competing ideas for reward assignment were evaluated experimentally in the each of the three domains (Chaser, Sheep, and Car). The comparison of these two alternatives was done for three runs, each consisting of 20 training sessions, using an initial random population. One human test subject, codenamed Green, was used as the instructor of all these initial experiments. A statistical comparison was done on each population of individuals. Student's t-test was used because of the small number of samples. This provided a fair way to calculate significance, which is listed as the "p-value" in each table. As a reminder, a p-value of <0.05 on a single-tailed test rejects the null hypothesis that the populations are the same. This then supports the hypothesis that populations are statistically different to a 95% confidence level.

In general, for IL, the agents trained with the Reward-During mechanism nearly always outperformed the agents produced with Reward-at-End. This was true to a statistically meaningful level in all three testbeds. These results lend initial credibility to the IL approach. One of the main reasons for the superiority of Reward-During is believed to be the number of updates to the population of individuals. Even though the individuals are evaluated at every time step, the Reward-During receives many more PSO iterations for training the network. Effectively, the Reward-at-End only does one iteration per training session. Each domain received 20 training sessions; thus, PIGEON only received 20 iterations for learning. For Reward-During, a PSO iteration would occur once a second during the run for the 20 iterations. The agents therefore received 600 iterations for the 30-s Chaser runs (20×30) and 1200 iterations for the 60-s duration of the other two domains (20×60). These additional iterations allowed each of the competing agents in a population to jostle for position more often, as the sorting for position happens during each PSO iteration step. This allows the

Table 1: Instructional Learning – Comparison of Two Methods on Chaser.

Instructor Code	Method	Mean PR	StdDev	p-Value
Green	Reward-at-End	22.90	2.89	0.0302
Green	Reward-During	28.52	1.31	

PR, performance rating; Max PR value=30.

Table 2: Instructional Learning – Comparison of Two Methods on Sheep.

Instructor Code	Method	Mean PR	StdDev	p-Value
Green	Reward-at-End	33.75	1.35	0.000104
Green	Reward-During	50.46	1.64	

PR, performance rating; Max PR value=60.

Table 3: Instructional Learning – Comparison of Two Methods on Car.

Instructor Code	Method	Mean PR	StdDev	p-Value
Green	Reward-at-End	32.26	9.97	0.0409
Green	Reward-During	50.84	1.35	

PR, Performance rating; Max PR value=60.

agents in the population to influence each other more during the PSO update. Additionally, because the feedback happens more often (once a second) for Reward-During, the attribution of which particular action is being rewarded could be easier for the agent to identify. For these reasons, Reward-During was selected as the method of choice for assigning credit in IL. Tables 1–3 indicate the results of the experimental evaluation. The higher the mean PR, the more successful the method is. Note that the difference is greater as the complexity of the testbed increases.

In the next two sections, we experimentally evaluate our concept of IL as implemented with PIGEON using a Reward-During learning strategy. In particular, we seek to

1. Determine whether it can stand by itself as the sole agent for building technique (Instructional only);
2. Determine whether it can improve/enhance agents when combined with other learning methods.

5 Instructional-Only Performance

Instructional-Only (IL-Only) learning creates agents from scratch, based solely on the interaction with a human instructor. The agents are initially random, and the instructors were charged with building correct behaviors in these random agents through IL. The experiments were designed to determine (i) whether an agent can be successfully taught solely through IL, and (ii) what level of proficiency could be attained by an agent produced through IL-Only learning. In this case, the level of proficiency was defined by the PR of the agent. Two human instructors, codenamed Violet and Orange, were each tasked with creating an agent from scratch using IL in Chaser, Sheep, and Car.

The scoring metrics of agents in all three testbeds can be found in Table 4. The column “IL-Only Best DR” represents the Didactic Reward of the agent that received the highest Didactic Reward. Didactic Reward can be likened to the similarity of the agent’s action to that of the instructors. Remembering that no action on the part of the instructor means full reward, the higher the reward means the most similar the behavior to that of the instructor. The column “IL-Only DR” represents the Didactic Reward of the agent that had the best PR, while “IL-Only PR” was the actual PR for that individual. Note that the maximum achievable values for DR and PR are 30 for Chaser and 60 for Sheep and for Car. It can be seen that the IL-Only agent that received

Table 4: Results for Instructional Learning: IL-Only.

Testbed	IL-Only Best DR	IL–only DR	IL-Only PR	Violet PR	Orange PR
Chaser	29.10	29.10	26.04	22.7	24.6
Sheep	51.54	43.18	25.35	29.7	43.5
Car	50.89	47.55	1.29	26.6	39.4

the highest Didactic Reward was not always the agent with the highest PR. This is not unreasonable because even expert instructors can be imperfect. The agent that most resembled the instructor's actions would not necessarily be the best when performing a particular task. The individual that received the most reward may not necessarily have been able to operate successfully on its own. We did not consider further the concept of instructor error, but leave that for future research. A larger number of instructors participating in the experiments may shed more light on this issue. Nevertheless, for the purposes of this paper, the system seeks to behave like the instructor, whether that is perfect or not.

An example population of Chaser can be seen in Figure 5. Note that in Figure 5, the Y-axis labeled "fitness" is the PR, with a maximum value of 30. The IL-Only system was able to steadily improve the best agent from an original set of random agents, and eventually receives close to full reward (a PR of 30). Because PSO forces learning on the entire population of agents, it can be seen that not only does the best agent get better, but the worst agent in the population also improves. Their PR improvements were found to be from 24 to near 30 for the best agent and from 13 to 27 for the worst. The entire population begins to converge to a single point near 30, even with the relatively few 20 instructional sessions held.

The initial random agents would simply "rattle the joysticks" in the beginning. Later on, however, as the training session progressed with IL intervention by the instructor, the agents would begin to follow the fleer when brought close. Then, the agent would smooth out its movement and follow the fleer more often, even when turned away. Finally, by the last session, the agent would typically follow the fleer from almost any initial position and require very little correction. One human instructor found that the "online, hands-on training [was] surprisingly responsive."

By looking at the "PR" column of Table 4, the IL-Only agents can be seen to clearly succeed in Chaser and perform somewhat competently in Sheep. The best-performing IL-Only agents were actually able to outperform both instructors on Chaser. The best agent in Sheep was approximately as good as instructor Violet, capturing seven sheep on average, although significantly worse than Orange. However, the IL-Only agents did not perform well at all on Car. This failure was likely the result of the higher complexity of this task.

This dichotomy could be endemic within the IL method itself. During the instruction of an agent for Car, the agent could be forced by the instructor to get to the middle of the road, after which the agent would autonomously follow the road. However, these agents, when operating on their own, could not properly make

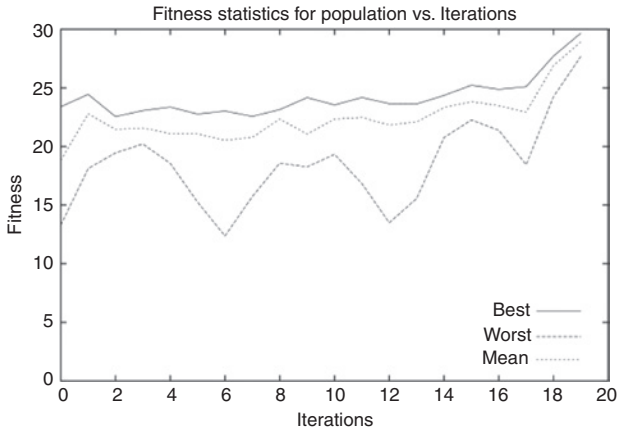

Figure 5: Instructional-Only on Chaser Testbed.

the initial left turn at the intersection, either hitting a car or missing the turn and driving off the track. The way our IL works, the agents were rewarded by the instructor for following the road and making proper turns at intersections. However, most of the training time is spent on the road and not at intersections. To optimize the reward, the learning algorithm would ignore punishment of actions that did not last long. We will seek to address this problem in future research.

In summary, IL alone was able to train an agent to an acceptable level in one of the three domains – Chaser. The results were mixed for Sheep, scoring close to one of the instructors. The method was effectively only shown the task 20 times, with a learning event once a second. In Sheep, although the PR was not particularly impressive, it was close to that of the instructors, making the agent as good as an instructor, which is what we ask of them. The IL-Only haptic-based training, however, failed to learn the more complex domain of Car on its own. Nevertheless, the advantages of IL were demonstrated by learning quickly and in real time using only a limited number of sessions.

6 Observational-then-Instructional Learning

As discussed earlier in this article, we hypothesize that IL could also be used to enhance the performance of an already existing agent. The purpose of this section is to determine whether IL can improve on the performance of agents built through other means (although still based on neural networks). One special type of supervised learning is where an agent is automatically created from observation of a human expert performer (called an *actor*) who performs a task in an environment (typically a simulation). Learning from observation (LfO) was used to initially create a competent agent by observing our two actors in each of the three testbeds. Then, after the LfO training was finished, the same actor/instructor was allowed to refine the agent by subjecting it to IL, and thereby adjusting it for errors and gaps thorough the haptic interface described above. The observational learning (LfO) was also done using PIGEON. We thus hypothesize that IL can be used to correct for flaws and fill situational gaps in an observationally trained agent.

Experiments were done to determine whether IL helped improve the performance of existing observationally trained agents. IL took place on agents in all three testbed domains, and their PR was then compared to determine whether the agent improved its performance, as measured by the PR, with the additional IL step.

Twenty IL sessions were used to improve the performance of these agents. The plots of the rewards over the 20 training sessions for each of the three domains are plotted below. It can be seen in Figure 6 that for Chaser, the observationally trained agents were already performing quite well (PRs of 28 and 29.5 out of 30, at 0 generations). This means that the original Observational-Only agents were quite capable, and the rest of

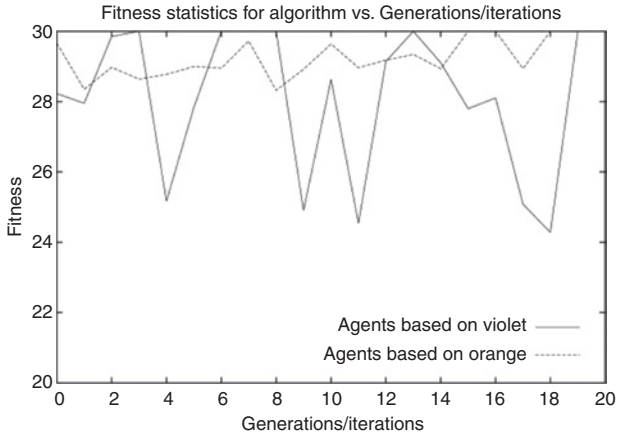

Figure 6: Observational then Instructional on Chaser Testbed.

the plot shows the improvement caused by IL. Of the two instructors, Orange was more evenhanded in correcting for small differences. Violet, on the other hand, tended to overcorrect the agent. Violet stated that "... sometimes the agent will make a mistake, which I would try to correct with unusual actions, which the agent picks up as a regular behavior." This relates to the quick learning rate purposely incorporated within IL. The agent trusts the instructor rather implicitly. Much of the rest of the training run was spent trying to undo what was unintentionally taught. In the case of Orange, the IL step was able to cause the agent to overcome these problems in Chaser.

In Sheep, the observationally trained agents also performed quite well from the outset. In this case, instructor Orange stated, "It's difficult to know when to correct since its action at any point in time could be part of a larger sequence of movements. Altering a particular action could invalidate an otherwise sound strategic situation." Orange actually felt he was making the agent perform worse with additional training. Conversely, Violet felt the agent was better than he was. These beliefs are confirmed in the reward plot of Figure 7 by the consistently higher Didactic Rewards being given by Violet.

In Figure 8, the reward for the Car domain had mixed results, with the rewards mostly distributed above 45.0 and a single case of 60.0, but without much consistency. On the one perfect 60.0 instance, Violet trusted the agent enough to completely allow it to control the car for the entire training session. This domain, because of its complexity, also caused some training problems. If an agent collided with another car, the instructor had to undertake some interesting maneuvers to correct the situation in real time, such as move in reverse down the road or pull off the lane until the traffic passed and then pull back into the lane. These behaviors were learned by the agent on a context-free basis, and sometimes detrimentally affected its performance.

Finally, the instructors noticed that in the Car simulation, the agent would sometimes behave erratically at the beginning of the run. This could perhaps be because the agents have not yet been sorted out properly during the start of the run, as all agents in the population are vying for control of the simulated car. If there is a tie at the beginning, an agent is selected at random, regardless of its performance. Therefore, because the first step in the behavior is to wait to merge into traffic, no feedback is provided by the instructor. However, after the first second of simulation time, the instructor would penalize all agents desiring to go forward, the population would re-sort, and the agent felt through the haptics interface would seem to begin to behave more sanely. This problem could be easily corrected by remembering the sorted list of agents from the end of the previous run.

The results in Table 5 are unfortunately inconclusive. While for some domains and instructors, there were improvements, these were minor and not statistically meaningful. Other domains saw some slight reductions in performance after IL. The IL phase made small improvements for Orange in all domains. The agents that learned from Violet, on the other hand, had decreases in performance for all domains. However, each of these was not different to a statistically meaningful degree ($p<0.05$) except for the Chaser domain. Therefore, it

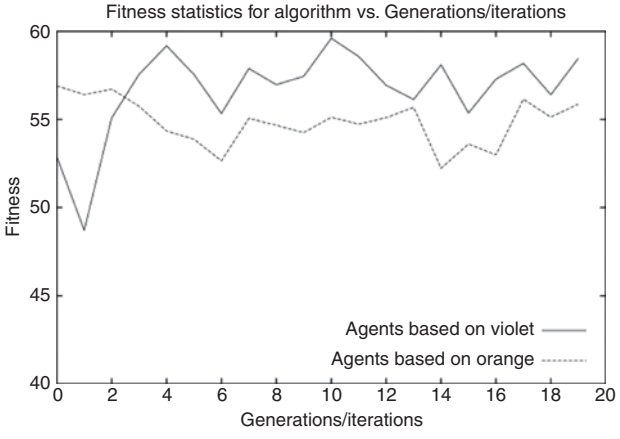

Figure 7: Observational then Instructional on Sheep Testbed.

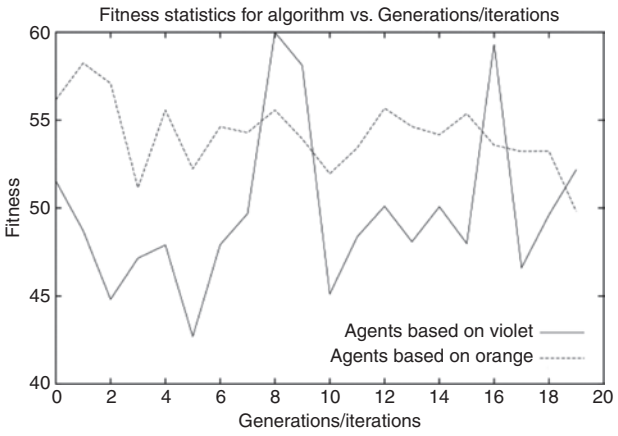

Figure 8: Observational then Instructional on Car Testbed.

cannot be concluded that IL improved an already good agent produced through LfO. The results also suggest that IL is highly dependent on the instructor; this is disappointing, but in reality no different from human-to-human training. If the IL improvement is based predominantly on the training abilities of the instructor, potential cross training of agents could be explored. We will seek to do further research with more human instructors to shed more light on this matter.

Furthermore, while the LfO-taught agents proved to be capable all on their own and thereby not showing much room for improvement, IL could conceivably be more useful when improving less capable initial agents. In fact, one could convincingly argue that IL-Only is nothing more than improving a very poorly behaving agent (random). Therefore, we argue that IL can serve as a tool for a good instructor to improve upon poorly performing existing agents.

In summary, the performance was within a 5.0% difference between the LfO-taught agents and the observation-then-instructionally-taught agents in the three domains. The agents did not appear to have made significant improvements through IL, but the method was generally not detrimental. We discuss this further in Section 8.

7 Qualitative Results

As a complement to the quantifiable results with Didactic Reward and PR, a qualitative analysis was done by means of a questionnaire. Each instructor filled out a subjective survey on the teaching experience in relation to the haptic training. The questions sought to draw conclusions about the perceived performance of the agent, the "intelligence" expressed by the agent, and whether the instructor believed that the force feedback helped in the training process. The answers ranged from 0 to 5, where 0 was absolutely negative while 5 was absolutely positive.

Table 5: Results for Observational-then-Instructional Training.

Simulation	Human	DR IL-Only	PR Obs-Only	PR Obs-IL	p-Value
Chaser	Violet	30.00	27.03	26.28	4.09e-23
Chaser	Orange	30.00	26.67	27.00	1.98e-26
Sheep	Violet	59.01	46.71	45.17	0.251
Sheep	Orange	56.01	44.84	45.09	0.456
Car	Violet	52.99	17.89	14.92	0.146
Car	Orange	52.89	31.48	36.48	0.0655

We acknowledge that a population of two instructors is much too small for us to make any scientifically valid assertions. However, we would have been foolish to pass up an opportunity to debrief them. Below, we report their comments and opinions as anecdotal evidence.

When asked, "Do you believe you could train the agent only haptically?," they responded positively with an average of 4.5 out of 5.0. Additionally, when the instructors were asked, "Do you feel the agent improved as a result of the haptic training?," both instructors responded positively with a 4.0 out of 5.0.

When the instructors were asked, "Did the haptics give a greater understanding of the agent's decisions?," they responded with a 5.0 out of 5.0. One trainer stated that "haptics does give an extra sense of what the agent is doing ... without the haptics, it would be more difficult to determine how to correct its behavior."

The instructors did find some flaws with the method. One instructor said that "... contradictory haptic feedback made movement and control that required accuracy and precision difficult." With the way the haptic interface was implemented, the agents' actions would still be felt even if the instructor wanted control. While the goal of the haptic interface was to feel the agent's intent, the instructor would have to correct for the movement with an additional disturbance. For example, if the instructor was trying to operate the car and the agent was still acting erratically, the joystick would try to move back and forth quickly, making it difficult to drive. A further study to determine whether this effect is detrimental to training for the instructor is warranted. However, that was deemed to be outside the scope of this research and is left for future research. Lastly, when asked "Do you feel that the haptics helped in the training?," they responded with a score of 3.5 out of 5.0.

Overall, the human instructors mildly agreed that haptics helped the training process. With some changes to the way in which the haptics-based training was applied, by reducing agent feedback while the instructor was in control, IL could be made to overcome some of the problems identified by the instructors and improve the agent training process.

8 Agent Validation

We noticed while performing the experiments that the Chaser agents produced through IL-Only (ultimately) received approximately 86% of the allowable didactic reward on average over the three domains, while the agents produced by Obs-IL received 95.5% and 93.8% for Violet and Orange, respectively. This implies that approximately this percentage of actions done by the agents was found to be acceptable to the instructors. This provides us with a potential means to do agent validation by the instructor. This could be a form of passive validation because not taking corrective action implies that the agent is behaving correctly. Moreover, the corrective action taken can be used by the system to correct whatever mistakes remained, thereby conceivably leading to a valid agent in one fell swoop. IL could also help the instructor create boundaries to identify any sections not found valid, and where the agents would need to be most urgently retrained.

Furthermore, by the nature in which IL is implemented, there is a built-in mechanism to sort and accept agents. Regardless of whether the agents were built from scratch or initially created through other means, IL deals with a population of agents all at once. One of the tasks of an experimenter is to select the best agent from that population to represent the final output of the system.

Clearly more research is necessary to formalize and quantify this concept of real-time passive validation. However, this was not our original intent, and we leave it for further research.

9 Conclusions

Throughout the literature, haptics has been said to be a means for high-bandwidth communication between humans and machines. We used haptic interfaces in our research as an additional medium that enables training intelligent agents by instructing them while they are performing a task or mission in real time. IL was designed to build agents from scratch, or to augment other learning approaches by providing a means to improve an agent's performance in real time.

9.1 IL as a Stand-alone Agent-Building Approach

The results of our Chaser experiments show that IL by itself could in fact teach a randomly acting agent to function acceptably in this relatively simple domain. The performance of IL-Only in the Sheep domain, while hardly excellent, showed some adequacy, especially in comparison with the instructor's performance. Nevertheless, it was remarkable that, by only using differences in force, an agent could be trained to perform acceptably well in real time after only a few reward sessions. Unfortunately, the Instructional-Only agent was not able to produce an acceptable agent in the more complex Car domain.

9.2 IL as an Means for Improving Existing Agents

Overall, the IL stage on top of the observational training stage did not show any meaningful performance improvements in the agents. The Observational-Only-trained agents performed close to the ability of the human instructors (Table 6). Of course, one could argue that the process that we call IL-Only is in effect an improvement of an already existing agent – one that behaves randomly. In retrospect, one limitation of our experiments was that the observationally taught agents that served as the initial agents to be improved through IL were already performing at a high level. This provided little room for improvement.

Nevertheless, the human instructors were able to gain insights from the force-feedback joysticks on what the agent was doing wrong and (inadvertently) provided validation of these already existing agents. The IL phase was meant to identify situations in which the agent had not properly learned what to do, and correct its errors. Because the agents are sorted according to the reward as awarded by the instructor, the agent with the most reward will rise to the top. The instructor, in effect, selects the best performer from the population, albeit indirectly. Because of this fact, the instructor is integral to this process and poor instruction can result in decreased performance.

9.3 Final Thoughts

The results obtained from our experiments were mixed. Our qualitative results indicated that the haptic interface was shown to help the training process by giving the human a better understanding of the agents' actions, and providing a mechanism to create and improve agents (as well as possibly validate their performance). However, the test sample was too small to make these conclusions general. The quantitative results indicated performance improvement in some experiments. However, most improvements were not statistically meaningful nor were they consistent. In fact, for one of the two human instructors employed in the experiments, the performance of the agents after application of IL was shown to actually degrade from the original observationally trained agents. This indicates that instruction is heavily influenced by the quality of the instructor – just as in human instruction.

In conclusion, while the results obtained did not generally support our hypothesis that IL through a haptic interface in real time could be advantageous for training tactical agents, we continue to believe that this is a very worthwhile concept to explore further. This is because IL was able to quickly create competent agents from scratch in relatively simple domains. This indicates the great potential for improvement of poorly

Table 6: Final Comparison Table.

Simulation	Best Human PR	Obs-Only PR	IL-Only PR	Obs-IL PR
Chaser	23.67	26.85	26.04	26.64
Sheep	36.62	45.78	25.35	45.13
Car	32.98	24.69	1.29	25.70

performing agents, at least in some domains. Haptics-based IL, therefore, seems to have some potential for being a stand-alone training approach. Lastly, IL through a haptic interface is thought to have great promise as an approach to the validation of agents.

10 Future Research

We hope to extend the work reported here by performing the same experiments with a larger number of human instructors. We also hope to test the concept of improving poorly performing agents through IL, rather than the already-highly performing agents created through observational learning. Other domains will be used in this future experimentation, some simpler than Chaser, others as complex or more complex than Car. Furthermore, there may be some advantages to be gained by contextualizing the tasks to be learned so that the learning can be done in context. We have had some preliminary positive results when context-based reasoning [52] is combined with PIGEON in observational learning for complex domains. This combination allowed agents to learn complex tasks that were not learned well through a context-free use of PIGEON. Lastly, the promising idea of passive validation should be explored fully. Validation of intelligent agents has been a problem in the past, and no workable solution is otherwise in the horizon.

It is possible that the performance could have improved if the instructors were given >20 sessions to work with the agents. However, 20 sessions seemed like a practical limit to avoid fatigue/boredom on the part of the instructors. However, we do not claim to have proof of this and leave it for future research.

Bibliography

[1] J. Aleotti, S. Caselli and M. Reggiani, Evaluation of virtual fixtures for a robot programming by demonstration interface, *IEEE T. Syst. Man Cyb. Part A.* **35** (2005), 536–545.

[2] T. Andou, Refinement of soccer agents' positions using reinforcement learning, in: *RoboCup-97: Robot Soccer World Cup I, LNCS 1395*, pp. 373–388, 1998.

[3] I. Arel, C. Liu, T. Urbanik and A. G. Kohls, Reinforcement learning-based multi-agent system for network traffic signal control, *IET Int. T. Syst.* **4** (2010), 128–135.

[4] C. Avizzano, J. Solis, A. Frisoli and M. Bergamasco, Motor learning skill experiments using haptic interface capabilities, in: *Proceedings* of the *11th IEEE International Workshop on Robot and Human Interactive Communication, 2002*, pp. 198–203, 2002.

[5] B. Bayart, A. Pocheville and A. Kheddar, An adaptive haptic guidance software module for I-TOUCH: example through a handwriting teaching simulation and a 3D maze, in: *IREE International Workshop on Haptic Audio Visual Environments and their Applications*, 2005.

[6] D. Bentivegna and C. Atkeson, Learning from observation using primitives, in: *Proceedings of the IEEE International Conference on Robotics & Automation*, Seoul, Korea, 2001.

[7] Y. Cai, C. Chui, X. Ye, Y. Wang and J. Anderson, VR simulated training for less invasive vascular intervention, *Comput. Graph.* **27** (2003), 215–221.

[8] S. Calinon and A. G. Billard, What is the teacher's role in robot programming by demonstration?: toward benchmarks for improved learning, *Interaction Studies* **8** (2007), 441–464.

[9] S. Chernova and M. Veloso, Confidence-based policy learning from demonstration using Gaussian mixture models, in: *Proceedings of AAMAS'07, the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2007.

[10] S. Chernova and M. Veloso, Teaching multi-robot coordination using demonstration of communication and state sharing, in: *Proceedings of AAMAS'08, the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Estoril, Portugal, 2008.

[11] J. Coelho, J. Piater and R. Grupen, Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot, *Robot. Auton. Syst.* **37** (2001), 195–218.

[12] H. Dinse, T. Kalisch, P. Ragert, B. Pleger, P. Schwenkreis and M. Tegenthoff, Improving human haptic performance in normal and impaired human populations through unattended activation-based learning, *ACM T. Appl. Perception* **2** (2005), 71–88.

[13] S. Eguchi, K. Sugiyama and Y. Wada, Robot task learning using haptic interface in virtual space, in: *SICE 2004 Annual Conference*, vol. 2, 2004.

[14] H. K. G. Fernlund, A. J. Gonzalez, M. Georgiopoulos and R. F. DeMara, Learning tactical human behavior through observation of human performance, *IEEE T. Syst. Man Cyb. Part B* **36** (2006), 128–140.
[15] M. A. Ferreira Crespo, W. Li and A. Gomes de Barros, Reinforcement learning agents to tactical air traffic flow management, *Int. J. Aviat. Manage.* **1** (2012), 145–151.
[16] D. Feygin, M. Keehner and F. Tendick, Haptic guidance: experimental evaluation of a haptic training method for a perceptual motor skill, in: *Proceedings of 10th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (Haptics 2002)*, Orlando, pp. 40–47, FL, 2002.
[17] B. Fisher, S. Fels, K. MacLean, T. Munzner and R. Rensink, Seeing, hearing, and touching: putting it all together, in: *International Conference on Computer Graphics and Interactive Techniques*, 2004.
[18] M. W. Floyd, B. Esfandiari and K. Lam, A case-based reasoning approach to imitating RoboCup players, in: *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society (FLAIRS)*, pp. 251–256, 2008.
[19] R. Gillespie, M. O. Modhrain, P. Tang, D. Zaretzky and C. Pham, The virtual teacher, in: *Proceedings of the ASME Dynamic Systems and Control Division*, pp. 171–178, 1998.
[20] M. Goodrich and M. Quigley, Learning haptic feedback for guiding driver behavior, in: *2004 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2004.
[21] D. H. Grollman and A. G. Billard, Robot learning from failed demonstrations, *Int. J. Soc. Robotic.* **4** (2012), 331–342.
[22] V. Gudise and G. Venayagamoorthy, Comparison of particle swarm optimization and Backpropagation as training algorithms for neural networks, in: *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03*, pp. 110–117, 2003.
[23] F. Guenter, M. Hersch, S. Calinon and A. Billard, Reinforcement learning for imitating constrained reaching movements, *Adv. Robotic.* **21** (2007), 1521–1544.
[24] C. Gunn, M. Hutchins, D. Stevenson, M. Adcock and P. Youngblood, Using collaborative haptics in remote surgical training, in: *Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems – Volume 00*, pp. 481–482, 2005.
[25] D. Haanpaa and G. Boston, An advanced haptic system for improving man-machine interfaces, *Comput. Graph.* **21** (1997), 443–449.
[26] A. Isaac and C. Sammut, Goal-directed learning to fly, in: *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington, DC, 2003.
[27] S. Kalyanakrishnan, Y. Liu and P. Stone, Half field offense in RoboCup soccer: a multiagent reinforcement learning case study, in: *RoboCup 2006: Robot Soccer World Cup X, LNCS 4434;* 72–85, 2007.
[28] J. Kennedy and R. Eberhart, Particle swarm optimization, in: *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, 1995.
[29] E. Klein, M. Geist and O. Pietquin, Batch, off-policy and model-free apprenticeship learning, in: *IJCAI-2011 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*, 2011.
[30] K. Kosuge, T. Fukuda and H. Asada, Acquisition of human skills for robotic systems, in: *Proceedings of the 1991 IEEE International Symposium on Intelligent Control*, pp. 469–474, 1991.
[31] A. Kucukyilmaz, T. M. Sezgin and C. Basdogan, Intention recognition for dynamic role exchange in haptic collaboration, *IEEE Trans. Haptics* **6** (2013), 58–68.
[32] J. Lee and S. Choi, Effects of haptic guidance and disturbance on motor learning: potential advantage of haptic disturbance, in: *2010 IEEE Haptics Symposium*, 335–342, 2010.
[33] X. Li and L.-K. Soh, Investigating reinforcement learning in multiagent coalition formation, in: *AAAI Workshop on Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems*, Technical Report WS-04-06, pp. 22–28, 2004.
[34] C. Madeira, V. Corruble and G. Ramalho, Designing a reinforcement learning-based adaptive AI for large-scale strategy games, in: *AIIDE Conference*, 2006.
[35] A. Merke and M. Reidmiller, Karlsruhe brainstormers: a reinforcement learning approach to robotic soccer, in: *RoboCup 2001: Robot Soccer World Cup V*, 2001.
[36] R. Mitchell, Kinesthetic-visual matching, imitation, and self-recognition, in: *The Cognitive Animal: Empirical and Theoretical Aspects of Animal Cognition*, pp. 345–51, 2002.
[37] L. Moriarty and A. J. Gonzalez, Learning human behavior from observation for gaming applications, in: *Proceedings of the FLAIRS 2009 Conference*, 2009.
[38] S. Niekum, S. Osentoski, G. Konidaris and A. G. Barto, Learning and generalization of complex tasks from unstructured demonstrations, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012*, pp. 5239–5246, 2012.
[39] S. Ontañón, J. L. Montaña and A. J. Gonzalez, A dynamic Bayesian network framework for learning from observation, in: *Conferencia de la Asociación Española para la Inteligencia Artificial – CAEPIA 2013*, 2013.
[40] J. L. Patton and F. A. Mussa-Ivaldi, Robot-assisted adaptive training: custom force fields for teaching movement patterns, *IEEE T. Bio-Med. Eng.* **51** (2004), 636–646.
[41] M. Ponsen, H. Muñoz-Avila, P. Spronck and D. W. Aha, Automatically generating game tactics through evolutionary learning, *AI Mag.* **27** (2006), 75–84.
[42] C. Sammut, S. Hurst, D. Kedzier and D. Michie, Learning to fly, in: *Proceedings of the ninth International Conference on Machine Learning*, Aberdeen, pp. 335–339, 1992.
[43] S. Schaal, Computational approaches to motor learning by imitation, *Philos. Tr. Biol. Sci.* **358** (2003), 537–547.

[44] T. A. Sidani and A. J. Gonzalez, A framework for learning implicit expert knowledge through observation, *Transactions of the Society for Computer Simulation* **17** (2000), 54–72.
[45] D. Silver, R. Sutton and M. Muller, Reinforcement learning of local shape in the game of go, in: *International Joint Conference on Artificial Intelligence*, 2007.
[46] L. Spector and J. Klein, Evolutionary dynamics discovered via visualization in the breve simulation environment, in: *Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems*, pp. 163–170, 2002.
[47] M. Srinivasan and C. Basdogan, Haptics in virtual environments: taxonomy, research status, and challenges, *Comput. Graph.* **21** (1997), 393–404.
[48] K. O. Stanley, B. D. Bryant and R. Miikkulainen, Real-time neuroevolution in the NERO video game, *IEEE T. Evolut. Comput.* **9** (2005), 653–668.
[49] K. Stanley and R. Miikkulainen, Evolving neural networks through augmenting topologies, *Evolut. Comput.* **10** (2002), 99–127.
[50] G. Stein, FALCONET: force-feedback approach for learning from instruction and observation using natural and experiential training, PhD Dissertation, University of Central Florida, 2009.
[51] G. Stein and A. J. Gonzalez, Building high-performing human-like tactical agents through observation and experience, *IEEE T. Syst. Man Cyb. – Part B* **41** (2011), 792–804.
[52] G. Stein and A. J. Gonzalez, Learning in context: enhancing machine learning with Context-Based Reasoning", *Applied Intelligence* **41** (2014), 709–724.
[53] G. Stein, A. J. Gonzalez and C. Barham, Combining NEAT and PSO for learning tactical human behavior, *Neural Comput. Appl.* (2014), 1–18, DOI 10.1007/s00521-014-1761-3.
[54] P. Stone and M. Veloso, Multi-agent systems: a survey from the machine learning perspective, *Auton. Robot.* **8** (2000), 345–383.
[55] G. Tecuci, Building *Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tools and Case Studies*, Academic Press, San Diego, CA, 2008.
[56] Y.-C. Wang and J. M. Usher, Multi-agent learning approach for developing routing policies in multi-agent production scheduling, *Int. J. Adv. Manuf. Technol.* **33** (2007), 323–333.
[57] S. Wender and I. Watson, Using reinforcement learning for city site selection in the turn-based strategy game Civilization IV, in: *IEEE Symposium on Computational Intelligence and Games, 2008. CIG '08*, pp. 372–377, 2008.

Bionotes

Gary Stein
Primal Innovation, 201 Tech Drive, Sanford, FL 32771, USA

Gary Stein received his PhD in computer engineering in 2009 from the University of Central Florida. He is currently employed by Primal Innovation Inc., in Sanford, FL, USA. His research focus is on machine-learning techniques to replicate or replace human operators in both simulated and robotic entities.

Avelino J. Gonzalez
Intelligent Systems Laboratory, University of Central Florida, 4000 Central Florida Boulevard, Orlando, FL 32816-2362, USA, **Avelino.gonzalez@ucf.edu**

Avelino J. Gonzalez is a professor in the Department of Electrical Engineering and Computer Science at the University of Central Florida, Orlando, FL, USA. His area of expertise is intelligent systems and machine learning. He received his PhD in electrical engineering in 1979 from the University of Pittsburgh.