VIRTUAL BATTLESPACE BEHAVIOR GENERATION THROUGH CLASS IMITATION

THESIS

Bryon K. Fryer, Jr., Second Lieutenant, USAF

AFIT/GCO/ENG/11-04

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/11-04

VIRTUAL BATTLESPACE BEHAVIOR GENERATION THROUGH CLASS IMITATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

Bryon K. Fryer, Jr., B.S.

Second Lieutenant, USAF

March 2011

AFIT/GCO/ENG/11-04

VIRTUAL BATTLESPACE BEHAVIOR GENERATION THROUGH CLASS IMITATION

Bryon K. Fryer, Jr., B.S.
Second Lieutenant, USAF

Approved:

[illegible]	10 MAR 2011
Lt. Col. Brett J. Borghetti (Chairman)	Date
[illegible]	10 MAR 2011
Maj. Michael J. Mendenhall (Member)	Date
Kenneth M. Hopkinson	10 MAR 2011
Dr. Kenneth M. Hopkinson (Member)	Date

AFIT/GCO/ENG/11-04

Abstract

Military organizations require realistic training scenarios to ensure mission readiness. Furthermore, developing the skills required to differentiate combatants from non-combatants is important for ensuring the Law of Armed Conflict is upheld. In Simulated Training Environments, which serve as a main resource for developing these required skills, one of the open challenges is to correctly simulate the appearance and behavior of combatant and non-combatant agents in a realistic manner. We show that a statistical learning machine can be used to observe the behavior of an agent and objectively determine if the agent's behavior is appropriate. Our approach is to first train the statistical learning machine by allowing it to observe thousands of iterations of agents performing what we define as desired behavior. Then we use the same learning machine to drive the behavior of a single new agent and compare the new agent's behavior to the prior observed behaviors. Following the comparison, the machine evaluates how close the agent behavior is to that of the desired behavior. Our construction of a data driven agent is capable of imitating the behaviors of the Virtual BattleSpace 2 behavior classes while also being configured to advance to a waypoint specific goal. The resulting agent supports the conjecture that combatant and non-combatant behaviors within simulated environments can be improved through the use of behavioral imitation. We are successful in creating this imitation agent with the ability to imitate four out of five classes of VBS2 Readiness Postures and also show that the incorporation of non-imitation based goals can be compatible with our agent's imitation abilities. Through these accomplishments, a new behavior generation tool is presented to training developers, which in turn, can be used to increase confidence in the accuracy and applicability of Simulated Training Environments.

Acknowledgements

First and foremost I am thankful for my fiancé and her unwavering support through this challenging endeavor. These last six years have been amazing; I can only imagine how awesome the future will be! ($\times\infty$)

I would like to thank the guidance and support of my advisor, Lt. Col. Borghetti, as well as my committee members. I would also like to thank the Air Force Research Laboratory's 711th Human Performance Wing for their sponsorship of this research as well as their facilitation of inspiring meet-ups and idea discussions. Finally, I would like to thank the Numerica Corporation and their incredible personnel for their technical support and extensive help.

Bryon K. Fryer, Jr.

Table of Contents

Page

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
AOR	Area Of Responsibility	1
FPS	First Person Shooter	2
CGF	Computer Generated Forces	3
STE	Simulated Training Environment	4
NPC	Non-Playable Characters	5
OaNC	Opponent and Non-Combatant	5
VBS2	Virtual Battle Space 2	21
API	Application Program Interface	21

VIRTUAL BATTLESPACE BEHAVIOR GENERATION
THROUGH CLASS IMITATION

I. Introduction

It was Combat Rescue Officer Lieutenant James Davis' second tour in South West Asia. This time, however, the stress was getting to him. During his last tour, his team had become acclimated to the rural and mountainous terrain indigenous to the region. His team had also become accustomed to the overt hostile identification that this sparse terrain had gifted to them. When his elite group of Pararescuemen, Combat Controllers and Special Operations Technicians, was inserted into a hostile Area Of Responsibility (AOR), the enemy was clearly defined and relatively easy to suppress. During this second tour, this luxury was replaced with a new found feeling of persistent reevaluation as their surroundings had been replaced by a dense urban environment, occupied by a disgruntled and tightly wound population. Davis and his team realized the imminent risk of civilian fratricide created by such a scenario. As the need for caution increasingly elevated in this complicated environment, day by day Davis came to feel that it was only a matter of time before tragic consequences befell his unit. Though trained to near perfection in his conscious abilities, Davis never dreamed that it would be his unconscious instincts that would be tested when his team was airlifted to the site of a downtown suicide bomber attack during their seventh week on station.

As a Combat Rescue Officer, Davis had been given the most advanced and extensive training that any service member could obtain. His physical abilities, combat knowledge, and leadership skills were honed to the pinnacle of human capacity. His mind was similarly trained and equipped with an extensive array of leadership and reactionary toolsets that allowed him to prevail in the most stressful of situations.

Many of these physical and mental preparations were conducted in real world exercises and training. When these exercises did not fully explore and hone his abilities, his training was supplemented with First Person Shooter-style (FPS) training. These types of simulations were especially useful to condition his recognition of and appropriate response to non-combatants and combatants within a hostile environment. Davis displayed exceptional performance in these training scenarios, owing largely to his hobby as a 'gamer' before his commission. Just as he could complete the entire *Halo* trilogy in one sitting, he could successfully complete series of training scenarios. His skills in this section of his training were recognizable even to the other trainees. It earned him the unofficial call sign of "Spidey," as his counterparts were often impressed with his ability to intuit the next actions and locations of the opposing forces and non-combatants within the training scenarios. Davis was also able to distinguish between the non-combatant actors and the combatants at a virtual distance that depicted actors as only a few pixels. Davis' recollections of his old call sign were promptly interrupted when the HH-60 pilot alerted the team that they were one-mile from the Landing Zone.

Davis and his team was the first unit into the AOR, fast roping into a maelstrom of chaos in the downtown market square. Their pre-brief indicated that the bomber had targeted a police recruitment location that was being protected by six Security Forces Airmen. Davis and his team also knew that, after the explosion, there was a report of additional gunmen firing near the landing zone. This information had been relayed to them by the only conscious airman, who was receiving treatment within sixty seconds of Davis' team hitting the ground. After the team had rallied at the cover position, they began the systematic process of securing the immediate perimeter. Davis' commands and actions were precisely timed as the team secured the area. It was then, as he was moving down a tight corridor, that approximately five feet ahead and to his left, a door flew open. A large, dark figure quickly emerged from the blackness of the doorway, and without looking, the figure began running down the corridor. The close proximity of the noise and the speed at which the figure appeared had caught Davis off guard. It was, however, not more than a quarter of a second later that he had his weapon targeted on the figure and another quarter of a second before his right index finger was softly pressed against the trigger. While time had frozen in this crucial moment, Davis had one clear thought resonating within his mind; that the outcome of this instant would forever change him. In this same instant, he also knew that he did not have the ability to escape this situation. He would be required to determine the fate of the figure, and there was not time to deliberate nor weigh any ramification of his dire choice. Before that seemingly infinite yet infinitesimal second reached its conclusion, James Davis had already made his decision.

1.1 Motivation

A FPS training environment, as used in Davis' training, places the trainee in the role of a single individual inside a simulated environment. The player controls this individual as if they were that person, with the goal of achieving tactical objectives as well as avoiding harm to their virtual representation. The artificial actors within that environment, known as the Computer Generated Forces (CGF) [15], aid in the creation of an artificial training scenario and provide the human element within the virtual world. CGFs also aid mightily in accomplishing the most important goal of any FPS environment; to create a high a level of immersion for the player. Immersion is the concept that when the situation and the realism of the environment are sufficiently accurate, the trainee will interact with and perceive the virtual world as if the same events were unfolding in the real world. Within these environments, the goal of player immersion carries increased weight. As a result of the training that is conducted within a Simulated Training Environment (STE), military members are expected to apply the knowledge and experience gained into real world situations. If the immersive aspects of the STE are lacking, the result can be improper training which could lead to unnecessary casualties.

The critical decision that Lieutenant Davis faced is not limited to career fields that lend themselves toward special operations and increased combat involvement. All deployed service members require the ability to make immediate and resolute decisions. In order to prepare service members with a skill-set designed to properly handle these instant reaction situations, our ability to facilitate appropriate and actionable training needs to be evaluated.

The current approach for attempting to fulfill this requirement relies on FPS environments. This training not only lacks in the ability to present quality immersive training environments, but it also lacks a way of measuring the events within the scenario with a believability scale that articulates a standard. Put simply, the state-of-the-art approach currently used to deliver training within a FPS environment is not measuring up to the requirements needed to facilitate effective and immersive military training.

1.2 Problem Description

Improving the realistic capabilities of FPS environments is an open area of research that is shared by academics, the entertainment industry, and the military [15]. Specifically, the military leverages advancements within this domain to improve training through the use of Simulated Training Environments. These new methods of training allow for the simulation of complex and otherwise prohibitively expensive live training exercises. While the benefits of simulation-based training are readily apparent, the requirements for realistic behaviors and the need for the ability to create them is an area of research that has received little attention. FPS agents that represent Non-Player Characters or Opponents (NPC) must appropriately replicate the behavior of a human in order to be useful as a CGF [21]. While there are various techniques used to accomplish this desired end state, there is a distinct division between the intended audience of the CGF and the goals of the application in which it resides. These audiences and goals roughly divide into two domains; the entertainment-based FPS Video Games and the training-based Military Simulated Training Environments.

For FPS environments that are primarily focused on entertainment, the dominant design objective is to create NPC behaviors that support a set of varying levels of difficulty that allow the consumer to tailor their experience. Failure to achieve immersion within an entertainment focused FPS environment can result in a temporarily dissatisfied customer and potential lost profits for the producer.

Failures are more costly within a military STE. When Opponent and Non-Combatant (OaNC) actors are expected to facilitate an immersive and realistic environment that presents more than various levels of difficulty, but rather delivers teachable objectives and real world context, failures are unacceptable. This is because one of the imperative outcomes is the applicability of the training in real world situations. If the OaNC agents behave in non-standard and unrealistic ways during a training exercise, the quality of the training will be degraded and improper training could lead to unnecessary casualties.

1.3 Contribution

This research effort focuses on improving CGF realism by utilizing a trained knowledge of behavioral patterns in order to influence agent interactions with the simulated environment. This is accomplished by acquiring a set of distinct behavioral patterns which are of interest to be learned, learning the representative features that encapsulate the behaviors, and finally, using those learned behavioral encapsulations to drive agent actions.

Given a specific context and observed behaviors within an environment, can agents be designed and implemented to utilize these provided behaviors and gained information in order to generate verifiable believable behaviors, ultimately improving the legitimacy and applicability of a STE? A further extension of this research question examines if using this behavior generation technique is also capable of achieving geographically situated goals while generating these behaviors that are behaviorally driven.

This research effort presents a new tool to STE and training simulation designers. It provides a new way of developing agents with realistic actions by incorporating representative behavioral patterns.

Our agent is capable of exibiting a distinct behavioral pattern in the simulated environment. Our evaluations indicate that it was effectively able to align its behaviors toward the representative features of a desired class of behaviors. Furthermore, we have found that this imitation ability was able to be utilized while achieving a geographic goal.

This research does not stand alone as a final prescription for a realistic Computer Generated Force in Simulated Training Environments. It does, however, provide a demonstration of the ability to imitate trained behaviors that are guided by appropriate recorded behaviors. This approach to agent creation facilitates an agent's ability to create verifiable behaviors. Subsequently, if used as an additional tool in the development of training for our servicemen and women, improves training, then it has accomplished its ultimate goal. Through this improved training, one could be confident that Combat Rescue Officer Lieutenant James Davis would be able make the correct decision.

The remainder of this document is organized as follows: Chapter II examines related approaches that have been applied within this problem domain, as well as examining uses of behavioral imitation within simulated environments. Chapter III details our approach to testing our hypothesis including data set generation, experimental design, agent creation and evaluation. Chapter IV details the results of our agent's performance and its impact. Finally, Chapter V examines the results of our tested hypothesis, its impact, and the prospective future works that are appropriate and achievable.

II. Literature Review

This research is an effort to develop a novel method to generate First Person Shooter Opponent and Non-Combatant behaviors appropriate for training scenarios. This chapter provides background information and related research efforts that focus on realistic behavior generation techniques. It also examines how current methods and the underlying performance mechanisms that support them are not appropriate for this effort. Section 2.1 begins this discussion by outlining the current standard approach to behavior generation, and then concludes by contrasting that with the demands of our problem. Next, Section 2.2 outlines the major methodologies that have attempted to create realistic agent behavior. Finally, Section 2.3 examines the framework of agent behavior generation through imitation.

2.1 Agent Behavior Generation

The goal of First Person Shooter Games and Simulated Training Environments are to create immersive environments. In pursuit of this goal, developers have often focused on visual realism through graphical performance [34]. While this pursuit is meritorious in and of itself, by creating a virtual world that is visually appealing and convincing, a complementary focus has arisen. This focus is the creation of realistic human-like entities. These human-like entities will be referred to simply as agents. Agents that are tailored to exhibit opponent and non-combatant actions within this type of FPS environment are also called Computer Generated Forces (CGF) [21]. The complete definition of a CGF further emphasizes that the agent must appropriately replicate the behavior of a human [21]. This CGF requirement will be the baseline for the definition of realistic behaviors with respect to an agent's behavior generation.

The concept of realistic behavior generation has only recently gained momentum within the academic and commercial realms. The current focus of these two industries is centered upon improving entertainment value [29, 30, 31]. While any new research is encouraging, appropriate acting agents are also needed for use in training environments. The reasoning and background behind current, entertainment focused research will be detailed in Section 2.1.1 followed by the requirements and background associated with the need for explicitly realistic behaving agents in Section 2.1.2.

2.1.1 Entertainment Focus.

When a developer sets out to create the agents that will populate their FPS environment, developers often channel their efforts toward the product user. If the end user is expected to approach the environment with the intentions of being entertained, this in turn dictates a specific agent development approach. In general, researchers and game developers consider in-game opponents to be entertaining when they are difficult to defeat [9]. Although this may always be true for advanced players, it is not so for inexperienced players; for these casual gamers, a game is most entertaining when it is challenging but beatable [29]. The culmination of the need for sophisticated in-game actors within the video game entertainment industry has become evident with the recent rise in and demand of multi-player focused game-play, where skill-ranked human players battle against one another in order to attain an appropriate level of difficulty. This is a significant indication that game players are underwhelmed by the current abilities of the behavior mechanisms that govern their computer-controlled opponents and ultimately prefer to seek out human-controlled opponents [23]. The improvement of the quality of agent behaviors, while preserving the characteristics associated with high entertainment value [29], is desired in the event that human-controlled opponents are not available or are not conducive to advancing the users' training [30].

2.1.2 Realism, Immersion, and Training.

The word 'realism' carries several denotations; within the context of simulated environments, realism is the form which follows the style of art and literature that illustrates the depiction of subjects as they appear in everyday life. In this way, agent behavior generation techniques are attempting to represent actors realistically by portraying their behaviors that is representative of the desired entities' behaviors within the real world. This behavioral realism, coupled with the extent to which a subject will believe that they are interacting with the real world, creates a higher degree of overall realism [5]. With increased behavioral realism and more effective immersion also comes an increased ability to achieve social influence. Put simply by Guadagno et. al: "[T]he more realistic the behaviors of virtual human representations ... the more they will influence individuals with whom they interact" [18].

These desires for realistic behaviors and social influence converge in a single desired goal within simulated environments: immersion. Immersion occurs when the simulated environment perceptually envelopes an individual in such a way that they cannot perceive nor detect the difference between that simulated environment and the physical world [18]. This goal is important because an immersed subject is more likely to feel that they are truly in the same environment with the simulated actors, as opposed to a subject that is not immersed. Ultimately, immersion facilitates training and improves the retention rate of information and context specific awareness [5, 18]. These advantages warrant high value in military training environments.

The Office of the Chief Scientist of the United States Air Force has emphasized this military value, and the need for improvement in the forward-looking Technology Horizons [22]. They cite the need for improved human behavior modeling and cultural behavior modeling as a part of advanced constructive discovery & training environments. This need, among a short list of others, is one of the "most essential Science & Technology for Air Force to pursue over the next decade" [22]. This expressed need motivates our research efforts; seeking to encapsulate and represent complex behaviors in a succinct feature space as well as facilitating the application of these models in improving training environments by providing trainers and trainees with a realistic behavior generation construct that is based in observation.

2.2 Traditional Behavior Generation Approaches

This section details the approaches that are used to generate agent actions within a simulated environment. These approaches range from Role Playing Games to traditional First Person Shooters. However, they all have a common goal: developing an agent to appropriately or realistically respond to dynamic environments. Each section's behavior generation solution details how the problem is approached, what the successes of the approach were, and the shortcomings and assumptions that limit the approach. It is also important to consider the ability of each technique to generate realistic behaviors in a highly dynamic environment. In order to examine these techniques that are striving for a realistic behavior, we compare the expected behaviors with the appropriate replication of human behaviors.

This type of examination stems from the idea of an *oracle* which has the ability to faithfully reproduce the correct and desired output independently of the system which is being evaluated [19]. This means that for each behavior generation technique that we consider, we can objectively evaluate its expected output with respect to the behaviors that a human would have displayed in the same situation.

2.2.1 Scripted Agents.

A 'script,' within the context of the agent development, is a finite set of instructions or actions that an agent executes in sequence [30]. These defined actions give the agent the full scope of behaviors which they can exercise within the environment. Scripted agents (agents whose main deliberation and action components rely on a script) are used to create a simplistic appearance of intelligence. Scripted agents are capable of interacting with their environment and achieving a defined goal.

Scripted agents are widely used to implement simulated opponents and non-combatant behaviors. Specifically, scripted behavior generation appeals to game developers because of its simplicity and directness. Their ability to directly dictate behaviors makes the agent understandable, predictable, adaptable to specific circumstances, easy to implement, easily extendable, and usable by non-programmers. Because of this accessibility, many researchers and game developers have shown that scripted agent behavior generation is applicable to many domains including Real Time Strategy, First Person Shooter, and Role Playing Games [35].

These accomplishments, however, are dwarfed by the shortcomings associated with scripted agent behavior performance. As mentioned previously, scripted agents are capable of achieving a defined goal and can interact with the environment as defined by their component scripts. As the definition of a scripted agent implies, these abilities are limited to goals and methods that have been pre-determined and dictated within the script. This static nature of scripted agent behaviors tends to increase their length and complexity as increased numbers of behaviors or more complex behaviors are added [7]. From static nature and complexity follow two issues that severely cripple scripted agents: adaptability and exploitability. Because the agent behaviors and reactions are pre-defined they cannot adapt to new environments. Therefore, agent behaviors that are selected within new environmental conditions may result in poor or incorrect behaviors. Similarly, if a human player acts in a way that the agent is not prescribed to handle, the player may be able to exploit the agent. This type of poor performance is desirable in neither the entertainment domain [9, 30, 31] nor the training domain [38].

When comparing scripted behavior generation technique with the behaviors of a real human, we would find that the agent is very capable of generating realistic behaviors for every situation that it has applicable scripts to execute (assuming that the script writer has taken up the resource intensive task of writing these scripts in a realistic manner). The scripted behavior generation technique would have a significantly lower realistic behavior generation ability given the entire set of possible situations. It is for this reason, coupled with the intensive resources required to write scripts for each situation, that a purely scripted approach is not generally suited for realistic agent behavior generation.

2.2.2 Machine Learning Agents.

From the shortcomings apparent in the widely used scripting techniques, Spronck et al. set out to incorporate machine learning techniques into the agent design, making it capable of adapting its behaviors [30]. This adaptive behavior generation, termed Dynamic Scripting, builds upon the scripted paradigm. Characterized by Spronck et al. as a stochastic optimization approach, dynamic ccripting seeks to find the most effective scripted behavior given its current environment. This is accomplished by attributing weights to a rule-base (the full set of scripted actions that can be selected from) in order to select the current script that controls the agent's behavior. The agent evaluates its rule weights based on the performance of each selection; increasing weights where the agent is successful and decreasing weights where the agent fails. This allows the agent to rapidly increase the appropriateness of the agent's actions, even in changing environments [30].

Spronck et al. and Dahlbom et al. have shown the successes of the dynamic scripting technique in Computer Role Playing (Neverwinter Nights) [31] and Real

Time Strategy [10, 11] environments respectively. Specifically, Spronck et al. demonstrated that their dynamic scripted agent was able to adapt its actions to defeat a static opponent. The process was also shown to achieve dominance over the static opponent faster than a randomly control method (Monte-Carlo). The dynamically scripted method of agent behavior generation ultimately adds diversity, adaptability, and appropriate difficulty to the opponent.

These additional attributes significantly improve an agent's ability to adapt its behaviors to environmental and human-player actions; however, the basic actions of the agent's behaviors need to be manually scripted. The agent behavior scripts that are created also require additional information, limiting the environmental attributes where each behavior is appropriate. Finally, if these conditional arguments are not correct, rule-base weights can be incorrectly affected if actions are taken that are inappropriate for its current state [30].

Due to these limitations, an agent's behaviors that were generated using dynamic scripting efforts display similar performance to scripting techniques. Contrast will occur in comparison to a traditionally scripted agent behavior generation because the dynamically scripted agent behaviors will behave realistically for a larger set of situations due to its adaptability. Ultimately, dynamic scripting is effective for optimization of the set parameters which dictate the weighting process. Unfortunately, the ability to optimize behavior selection does not incorporate the ability to replicate the behaviors of a human actor outside of the manually dictated scripted behavior set.

2.3 Behavioral Generation Through Imitation

Each traditional approach to agent behavior generation as outlined in Section 2.2 has been limited in its ability to produce realistic behaviors purely by algorithmic means. These traditional approaches were often a focus in modeling an actor's behaviors based on the cognitive processes that underlie their actions and differentiations that distinguish basic and advanced skills (opponent difficulty tailoring). A contrasting approach for modeling agent behaviors focuses on modeling the actor's behavioral patterns, preferences, or representative characteristics, called behavioral imitation [37]. This section looks at methods that have been put forth to accomplish this type of user modeling by incorporating a dataset of known good behaviors. This type of imitation-based agent behavior generation is accomplished by running an induction algorithm over the traces of good behaviors [2]. This process has been applied to areas of research like robotics [2], cognitive science [8], and user modeling [37]. Only recently, however, have these imitation techniques been adapted to simulated agents [1]. This type of behavioral imitation has distinct benefits and challenges, which are outlined below.

One starting point for approaching and understanding behavioral imitation is to begin by explaining what it is not. Behavior imitation approaches do not focus on creating a strict clone of the training behaviors. Like the difficulties exhibited by purely scripted agents, these rule sets would need to be fully exhaustive in order to be robust, despite changes in initial conditions and novel environments [2]. For that reason, we distinguish behavioral cloning from behavioral imitation.

In order to define what behavioral imitation actually means within the context of simulated environments, it is appropriate to look outside of the simulated agent domain and consider the work in robotic imitation by Bakker and Kuniyoshi [3]. Bakker and Kuniyoshi defined imitation as the event in which an agent learns of a behavior through the execution of that behavior by a teacher [3]. Through this process of learned behaviors, the agent is now capable of executing a behavior that it previously may have not known was possible or applicable. Bakker and Kuniyoshi display the benefits of this additional ability by illustrating three main areas that behavioral imitation helps improve.

First, an agent immediately begins learning behaviors that are likely to be effective. Specifically, by observing other agents' actions, they are learning behaviors that are already being used by agents successfully operating in the same environment [3]. Next, they indicate that the medium for learning is a common ground that has very low entry requirements. Agents can learn behaviors from each other without having any similar software, hardware, or shared communication mechanism because the communication "takes place at a high level, in terms of actions" [3]. Finally, Bakker and Kuniyoshi indicate that this new incorporation of behavioral imitation can operate seamlessly with other existing learning schemes. The resulting behavioral imitation can serve as an exploratory function of the problem domain while other learning schemes can then optimize the agent's performance.

With our exploration of behavioral imitation thus far, it would seem that behavioral imitation only lends itself toward replication of instantaneous behaviors. This notion is challenged by Thurau and Baukhage by performing behavioral imitation on all three levels of agent behaviors: Strategies, Tactics, and Reactive Behaviors [33].

2.3.1 A Behavioral Imitation Framework.

Both Bakker [2] and Bain [3] have put forth frameworks for behavior imitation and both are in general agreement as Table 1 depicts. They also agree that before approaching the framework, the agent design must define the goals and how the agent should evaluate and optimize the available behaviors. Each defined framework can be seen in comparison in Table 1.

This framework has been utilized in effective behavioral imitation agents. Their successes cover a wide array of domains including robotic behavior learning [3], ad-

Table 1. Behavior Imitation Frameworks

	Bakker and Kuniyoshi [2]	*Bain and Sammut* [3]
1.	Learn the actions that achieve the goals	Observe the action
2.	Construct high-level features from training behaviors	Represent the action
3.	Automated learning and interaction	Reproduce the action

vanced FPS opponents [33], auto-pilot agents [2], as well as Robo-Soccer agents and coaches [1, 27].

2.3.2 Limitations of the Framework.

The Behavior Imitation Framework does have limitations. As Webb et. al. outline, behavioral imitation induces drawbacks [37]. The first of these is the need for large datasets. The need for large datasets has been noted in many of the above agent designs and has, for some, been a limiting factor for the abilities of the agent. For example, the behavior capturing component of the Robo-Soccer agent, developed by Aler et. al. was not robust and user-friendly enough to gather a large dataset [1]. This then limited the abilities of the imitation behaviors. However, large dataset requirements are understandable. As Valiant has noted, any sufficiently difficult problem domain will result in learning algorithms that require many training examples to be accurate [36].

The next concern for behavioral imitation is the need for labeled datasets. Labeled datasets are needed because supervised machine learning explicitly requires labeled data while behaviors that are manifest to the agent may be without readily apparent labels. Webb et. al. also present three typical solutions for this need: explicit labeling (which may be prohibitively expensive), inferences based on logical clues (which require simple and label-able actions), and using a small set of labeled data to seed the behavior pool (which would be generalizable to the larger set of behaviors).

2.4 A Motivating Domain

The examination of related research has shown that there is a noticeable gap in traditional realistic agent generation techniques. In order to bridge this gap, a new form of agent generation has been found in Behavior Imitation. This new paradigm has proven to be capable of motivating effective performance. However, it too lacks a strong appeal toward explicitly realistic agents. While the Behavior Imitation Framework has its own difficulties, it provides a strong foundational design for the creation of explicitly realistic agents, explored further in Chapter III.

III. Methodology

3.1 Problem Definition

Non-combatant and opponent behaviors developed for use in Simulated Training Environments (STE) are not currently built with the explicit goal of realistic or believable behaviors. Within the STE, *believability* is the measure of how well an agent performs an action or series of actions in relation to how that action is performed by the same (intended) entity within a real world environment. *Verifiability* is the ability to quantitatively measure correctness. As discussed in Section 2.3.1, measuring and evaluating believability remains an open area of research. Working within these limitations, and in an effort to improve believability in an empirically measurable way, we focus on incorporating verifiability into our agent behaviors. Throughout the rest of this research we call this focus *Verifiable Believability*.

With this focus defined, it is appropriate to outline our research question: Given a specific context and observed behaviors, can agents be designed and implemented that create behaviors that explicitly achieve verifiable believability? Furthermore, is this ability compatible with a simultaneous objective of achieving geographically oriented goals? For the purposes of this experiment these geographically oriented goals are defined by traversing an area of interest from a start location, and completing the desired goal by arriving at a destination location.

The remainder of this chapter follows the Behavioral Imitation Framework outlined in Chapter II. Section 3.2 outlines the observation and representation portions of the framework (Table 1) as well as the behavior production portion of the framework. This examination of behavior production concentrates on the agent's design and how it makes use of the previous portions. Finally, Section 3.3 details the evaluations that examine the effectiveness of each component of the behavior imitation framework.

3.2 Problem Approach

In order to create an agent that is capable of achieving the objectives set-out in our research question, three main efforts are outlined. In lieu of real world data (that is logistically prohibitive to gather for the scope of this proof of concept research), synthetic behaviors are created, recorded, and labeled. A feature based classifier is developed and trained on these synthetic behaviors. Once the classifier has been established and trained, it is incorporated into the agent's behavior generation mechanisms. The following sections describe each of these efforts.

3.2.1 Observation of Behaviors.

In order to generate synthetic behaviors for use in conjunction with the classifier training process and subsequently within the agent's behavior generation mechanisms, we must consider the selection, design and creation of the environment where these exemplar behaviors are generated, how they are generated, and finally, how they are captured and labeled.

3.2.1.1 Simulated Training Environment - Virtual BattleSpace 2.

The Simulated Training Environment (STE) is the environment in which synthetic agents and our new agent perform their actions. It includes obstacles, enemies, friendly forces, civilians, and all other environmental attributes that are perceivable by the agent throughout the course of the scenario (e.g., the weather conditions). The environment dictates the complexity of the exercise and is appropriate in its scope for the desired training purposes. It is important to keep the STE environment constant throughout the classifier training, agent training, and agent evaluation phases in order to reduce or remove any unaccounted-for behaviors which may affect performance at any of these stages.

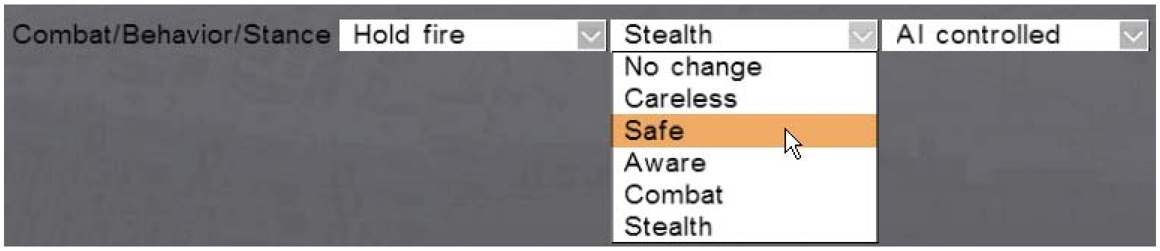


Figure 1. VBS2 Agent Behavior Selection

For the purposes of this research, we use the Virtual BattleSpace 2 (VBS2) software suite as our STE. VBS2 is a commonly used training suite within many military organizations including the United States Army [28], the United States Marine Corps [20], as well as the United States Department of Homeland Security [14]. It also has an advanced Application Program Interface (API) that allows for direct access to, and manipulation of, actors within the environment. This allows recordable simulations as well as advanced agent creation. The VBS2 military units have a set of selectable readiness postures which facilitate the synthetic behavior generation. Figure 1 shows the VBS2 Agent Properties selection (for this example, the “Safe” agent), as well as the other possible Behavior Classes: Careless, Safe, Aware, and Combat. Each of these Behavior Classes are described in detail in Appendix B. This labeling process, while simulated in our experiment by VBS2 readiness postures, can be accomplished on real world traces recorded from real world simulated training scenarios, real world engagements, virtual reality simulated training scenarios, or any other environment with behaviors of interest that are to be imitated.

3.2.1.2 Behavior Capturing and Labeling.

It is important to have the ability to record and analyze the behaviors of agents within the STE. This functionality is needed for the analysis of both the synthetically generated agent behaviors as well as our new mimicking agent. Recording is accomplished through the use of a VBS2 plug-in which records the instantaneous attributes of each agent present in the simulation at 1 Hz. The attributes recorded include the x, y, and z distances from the origin of the map (translatable to East, North, Up coordinates), the orientation, and the velocity vector of the agent. These logs are parseable and the attributes of each agent are discernable based on unique agent IDs.

Each waypoint that the agent is tasked with navigating is represented in one log file, beginning when the agent embarks and ending when the agent arrives at the goal waypoint. Finally, as each waypoint that the agent of interest is tasked to navigate to is logged, the resulting log is added to the agent of interest's Behavioral Class set of logs. These labeled logs are used in the classifier training process detailed in Section 3.2.2.

As a STE, VBS2 lacks the methods for automated data collection. In order to accomplish this desired function within the STE, a number of additional components that facilitate this process are incorporated into the behavior capturing and labeling process. First, the logging process is facilitated by a logging plug-in that was developed by the Numerica Corporation [25]. Next, the agent of interest is created within the environment, initialized to the desired readiness posture, instructed to hold-fire on any combatants, and logging started. This is accomplished through the use of VBS2 scripting commands and the SimExec Agent. The SimExec Agent serves as a bridge between the VBS2 environment and the other agents agents [24]. These other agents, the SimExec Agent, and our new agent are developed within the Jack Agents Development Environment (an agent oriented development environment which is built on top of, and integrated with the Java programming language). After the agent has successfully traversed the area of interest, it is removed from the simulation and the logging plug-in is instructed to end the log. This process is repeated to collect data for every scenario. Overall, this process allows for a large number of sample behaviors to be observed in a fully autonomous manner.

3.2.1.3 VBS2 Behavior Classes as Synthetic Behaviors.

The readiness postures that VBS2 units exhibit serve as the classes that represent the range of behaviors that make up the training data. Training data are labeled instances of real or simulated behaviors used to train a classifier. The training data contain the full set of population traces that are labeled according to the behavioral class dictated by the readiness posture.

These postures have desirable components that make them an appropriate choice for our proof of concept research effort. Locations and actions of the agents can be recorded each time step in VBS2. The actual process of recording the agents is detailed in Section 3.2.1.2. The available selection of behaviors also have a known reaction to environmental factors. For example, an agent with the posture of "Stealth," given a destination waypoint and a combat stance of "Hold Fire," upon encountering an opposing force, assumes a prone position and avoids contact with the target, low crawling to the destination.

3.2.1.4 Scenario Design.

The Synthetic Behavior Generation Scenario is designed in such a way that the desired outcome is a feature rich set of behaviors that display a large subset of the VBS2 Behavior Classes' available behaviors. For this scenario, we choose an area of interest to observe behaviors within a square grid of a VBS2 map (Figure 2, Left). The map is a representation of The Ohio State University campus which includes a diverse set of terrain [12]; however our area of interest is a selection from a section of this map that has a consistent geography, free of obstacles, and empty of any agents or game objects outside of those described below (shown in Figure 3). This simplicity decreases the complexity of the VBS2 Behavior Classes' interactions with the environment and narrows those interactions to those objects and actors added with intention. A detailing of the locations used for this scenario are found in Appendix D.

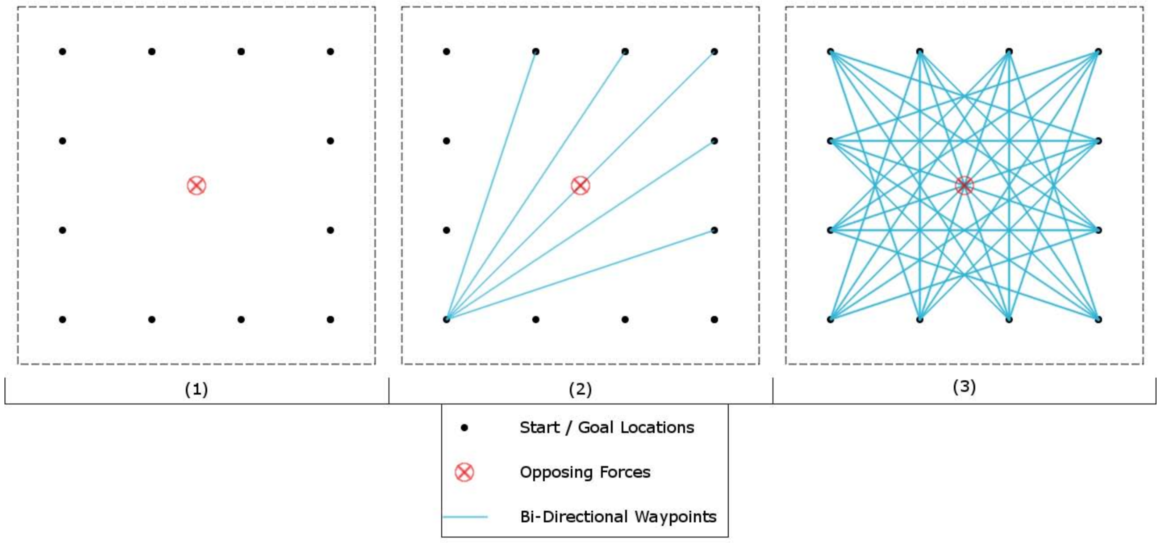


Figure 2. Behavior Generation Scenario. (Left) Scenario Setup. (Center) Single Start Location Traversals. (Right) Full Traversal Saturation

The opposing forces in the scenario are represented by one agent at the center of the area of interest. These agents are assigned a Combat stance of "Hold Fire." They are also given a command to remain at the same position throughout the data collection to ensure that their presence remained constant between each iteration. Figure 2 (Left), shows the location of the Opposing Forces.

Next, the agent's starting locations and goal locations are defined. Each of the solid black circles shown in Figure 2 represent a starting location and a goal location and are located in a square grid around the area of interest. Each starting location has five possible goal locations and is a goal location for five other starting locations. At each starting location, the corresponding goal locations are discovered by selecting all of the locations that are greater than three locations away from the starting location

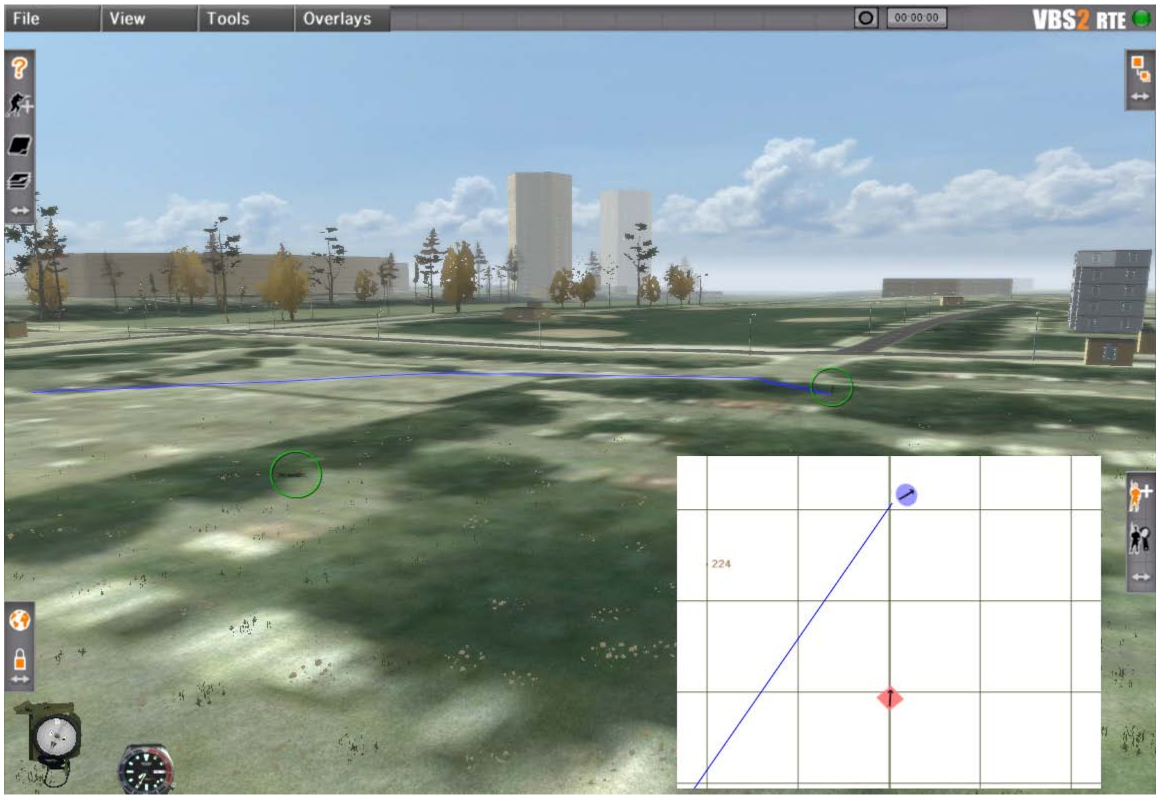


Figure 3. VBS2 Area of Interest

(as shown in Figure 2, Center). This selection process ensures that the resulting paths require interaction between the agents and the opposing forces. After each goal location has been determined for each starting location, the area of interest has a set of possible paths represented by solid lines in Figure 2 (Right).

Each VBS2 Behavior Class is directed from a start location to its corresponding goal location a total of 2040 times. For each of the traversals of the area of interest, the Behavior Capturing and Labeling process, detailed in Section 3.2.1.2, is accomplished.

3.2.2 Representation of Behaviors.

The feature generation and classification portions of our approach form the foundation of the contribution of representing and learning Opponent/Non-Combatant behaviors. Section 3.2.2.1 presents the feature-generation method that strives to capture the essential identifying and distinguishing aspects of agent behaviors. These features are then learned as training behaviors using a distance-based classifier, detailed in Section 3.2.2.2.

3.2.2.1 Feature Generation.

Position and orientation are recorded at a defined interval of 1 Hz and represent the raw behavioral data. The record of an individual agent over the course of a defined action is then reduced to a single point in multidimensional space. This multidimensional space is defined by the derived features which have been shown to encapsulate the behaviors. The derivation of these features and their ability to encapsulate an agent's behaviors has been outlined and evaluated in Appendix A.

3.2.2.2 Classification.

Our classifier is a supervised learning mechanism is used to identify a sample from a population and assign it the correct label. Within our research, the classifier processes a feature space sample and labels it according to the process which generated the sample. In order to accomplish this functionality, a classifier is trained and evaluated by presenting it a sub-set of the labeled population data and querying it on the remaining unseen data. This process is called k-fold cross validation [13]. Our classifier is presented with this training set, which is comprised of the samples created by the feature generation outlined in Section 3.2.2.1. The classifier is trained and evaluated by utilizing a leave-one-out cross validation. This training and evaluation process is identical to the classification process outlined and evaluated in Appendix A.

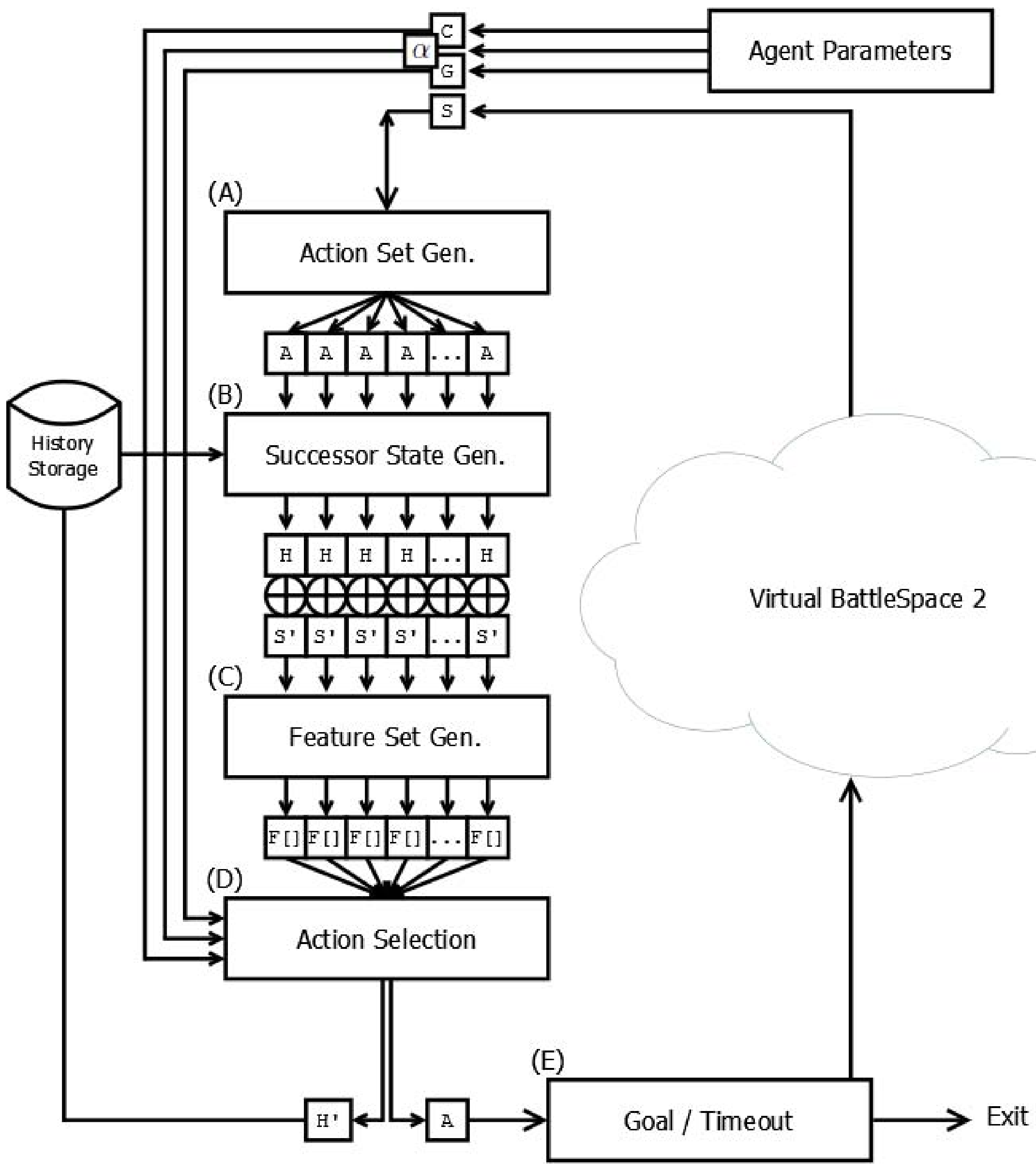


Figure 4. Behavior Generation Agent Model

3.2.3 Reproduction of Behaviors.

Equipped with the trained classifier generated through the feature generation and classifier training process outlined in Section 3.2.2 and supported by the scenario design and behavior capturing detailed in Section 3.2.1, we now describe the agent creation process. The goals of the agent are to travel to a waypoint while behaving in a manner that is indistinguishable from a specific class of behaviors. Within our specific experiment, the goal is to cross the area of interest and to behave as a specified VBS2 Behavior Class. The agent accomplishes this through the use of a utility-based agent model that incorporates classifier action evaluation. It is important to note that, although the readiness posture can be set in the VBS2 Agents while generating training data, no such ability exists in the agents created for this research. This processes shown in Figure 4 and described incrementally in the following sections.

3.2.3.1 Action Set Generation.

The agent first generates a set of possible actions from which it can select the best to execute. An action is a tuple of velocity and orientation, where velocity is drawn from 1, 3, and 5 m/s, and orientation is drawn from orientations at thirty degree intervals beginning at zero. This results in a set of 36 possible orientation-speed combinations. This sets of possible velocities and orientations are seen in Table 2

Table 2. Action Set Parameters

Velocity (m/s)	1, 3, 5
Orientation (degrees)	0°, 30°, 60°,90° 120°, 150°, 180°, 210° 240°, 270°, 300°, 330°

These ranges and increments were chosen in order to keep the evaluation and action portions of the agent to less than one second. This computational time limit is important to our agent's approach because the action and logging intervals are set to 1Hz. For simplicity, the agent does not take into consideration objects or opponents within the environment and simply generates this set of possible actions for future review. The resulting orientation-speed combination $[v, o]$ is represented as A_i where i is one of the k possible actions for a given current state S. A given state is represented by the location, time, and orientation $[t, x, y, o]$ of our agent. The resulting action operation is represented as

$$A_i \times S \rightarrow S'_i \tag{1}$$

This process takes place in component (A) in the agent model detailed in Figure 4, where S is the current state input from VBS2 to the Action Set Generation component.

3.2.3.2 Successor State Generation.

Once the set of orientation-speed combinations has been generated, these actions are evaluated and the resulting deterministic agent attributes are calculated. This is accomplished for each by applying the velocity and orientation combinations to the current state S in the manner shown in Equation 1. The resulting n states are delineated as S'_i where i is the same index of k that identifies the action A_i k possible actions.

The agent must be able to keep track of its past states. In order to accomplish this, a history vector is denoted as H. For example, a history consisting of n states would be represented as $H = [S_0, S_1, S_2, \ldots, S_{n-1}]$.

Each of the generated actions are then concatenated to a new temporary history H'_i which is defined by the concatenation ($\oplus$) of S'_i to H. The resulting history is identified by i as it uniquely contains the new state S'_i.

$$H'_i = H \oplus S'_i \tag{2}$$

An example concatenation is shown in Equation 3.

$$H'_i = [S_0, S_1, S_2, \ldots, S_n] \oplus S'_i = H'_i[S_0, S_1, S_2, \ldots, S_n, S'_i] \tag{3}$$

The resulting n potential actions, now represented as histories, can be passed to the feature set generation component of the agent. The Successor State Generation component of the agent is illustrated as component (B) in the agent model detailed in Figure 4.

3.2.3.3 Feature Set Generation.

Once the successor states have been generated and recorded in the temporary history vectors each history can be mapped to a corresponding feature set. The agent's behavior history, including the new additional action that is being evaluated, is submitted to the feature generation mechanism detailed in Section 3.2.2 and Appendix A. The resulting multidimensional point that represents these behaviors is represented as F and the feature generation function is represented by $f(\ )$. Using this nomenclature, we can define the newly generated features F_i for the i^{th} action as:

$$F_i = f(H'_i) \tag{4}$$

The resulting n potential actions and their resulting histories, now represented as feature sets, can now be passed to the action selection component of the agent. The Feature Set Generation portion of the agent is illustrated as component (C) in the agent model detailed in Figure 4.

3.2.3.4 Action Selection Through Utility.

The last step in the agent's decision making process is to select which action to execute. This selection is evaluated in both feature space as well as simulated space. The previous section has presented a set of n feature sets, each derived from an action A_i. In order to determine which action is most behaviorally correct, we must consider the trained classifier which will facilitate the feature space selection of the appropriate action.

The desired class that the agent should represent is passed into the agent through the Agent Parameters, as illustrated in the agent model in Figure 4 by the 'C' parameter. This parameter represents the trained class centers that were learned from our generated data in Section 3.2.2.2. Each class that is learned is denoted by C_j, where the particular class we would like to imitate is noted by j. As indicated in Section 3.2.2.2 these classification centers are an m-dimensional vector representing the point in feature space which is the center of the class' distribution.

Using the Euclidian distance between each feature in F_i and the classification center C_j, we can determine which action is most similar to the desired class j in feature space. This is calculated in Equation 5 for a feature space that contains m dimensions. Function designation dF represents the distance in feature space and each of the m dimensions are denoted as the second subscript of F_i and C_j.

$$dF(F_i, C_j) = \sqrt{(F_{i1} - C_{j1})^2 + (F_{i2} - C_{j2})^2 + \cdots + (F_{im} - C_{jm})^2} \tag{5}$$

This Euclidian distance measurement is also used in simulation space to determine the distance that the agent will be from the goal G at state S_i'. Function designation dS represents the distance in simulation space. This goal location is passed into the agent through the Agent Parameters and is illustrated in the agent model in Figure 4 by the 'G' parameter.

$$dS(G, S_i') = \sqrt{(G_x - S_{ix}')^2 + (G_y - S_{iy}')^2 + (G_z - S_{iz}')^2} \tag{6}$$

Using these two distance measurements, we can now define a utility function that will incorporate feature and simulation space in action selection. This results in a utility function for each action that is evaluated as:

$$U(F_i, S'_i) = \alpha \left(\frac{1}{dF(F_i, C_j)} \right) + (1 - \alpha) \left(\frac{1}{dS(S'_i, G)} \right) \tag{7}$$

The only component of this equation that remains undefined is α. The α value and its complement, $(1-\alpha)$, are set in order to determine how much utility is derived from the goal completion versus the ability to imitate the desired class. This parameter, set to values between 0 and 1, allows for the analysis of the tradeoff between behavioral imitation and goal completion, however, through the course of an agent's execution, this parameter will be fixed. The chosen alpha level is passed into the agent through the Agent Parameters. This is illustrated in the agent model in Figure 4 by the α parameter that is passed to the Action Selection portion of the agent.

After calculating each utility value for all of the possible actions, the action with the largest utility is selected. This action is then passed to the Agent Action portion of the agent. The selected history H' is also passed to the agent's history storage as indicated in Figure 4. The Action Selection Through Utility portion of the agent is depicted as component (D) in the agent model detailed in Figure 4.

3.2.3.5 Agent Action.

The final step is for the agent to pass the action that has been chosen to the environment and for this action to be executed. This is accomplished through the SimExec Agent that is also running simultaneously with our agent and serves as the link between our agent and the VBS2 environment.

3.3 Evaluation

As a proof of concept, we evaluate the approach using three measure of performance. First, the training set of recorded actions is evaluated to determine the classifier performance. This will give us an indication the performance we can expect an agent that uses this mechanism to drive its actions. This evaluation is outlined in Section 3.3.1. Next, we subject the agent to trials of waypoint traversals identical to those used to generate the training data. The agent executes actions until it has reached the goal waypoint or a time limit has expired. These events are triggered by the time and location values stored in the agent state. For the purposes of this research, the time limit that the agent has is set to one and a half times the maximum traversal time that was observed in the training data generation (80 seconds). After a trial has completed, the resulting behaviors are evaluated. This is accomplished for varying levels of α used in the agent's utility calculation. These evaluations are described in Section 3.3.2 and Section 3.3.3

3.3.1 Classifier Performance Evaluation.

Once the classifier has been trained on the agent behaviors, its ability to discern between the feature samples requires evaluation. In order to examine this ability, the classifier is evaluated with leave one out validation. This process is identical to the evaluations conducted in Appendix A. The result of this evaluation is a confusion matrix which shows the rates at which the classifier correctly identifies a sample as the correct class, as well as the rate at which the classifier improperly identifies the sample as the incorrect class. This evaluation is important to the agent evaluation because the agent's ability to generate imitation behaviors is limited by the classifier's ability to distinguish class samples.

3.3.2 Agent Imitation Evaluation.

The actions that the agent executes are evaluated by classifying the agent generated traces using the training data set trained classifier. These traces are generated by directing the agent to traverse the area of interest in the same way that the training data was generated, with the exception that the agent is directed to execute these traversals five separate times, each time imitating each VBS2 readiness posture. Furthermore, the classification of the agent takes place with the full set of generated truth data as the training set and the newly generated agent traces as the test data. The evaluation of these traces includes the feature generation and classification of each new trace as detailed in Section 3.2.2 and Appendix A. For this initial experiment, the agent is given a goal waypoint. However, it also is given an α setting that removes the goal distance from the utility calculation ($\alpha = 1$), as seen in Equation 7. This allows the agent to exist within the environment with its only goal being to imitate the desired classes. If the agent fails to reach the goal before a pre-determined timeout period has transpired, its execution terminates and the next iteration begins. The expected results of this evaluation is a classification accuracy similar to the classifier evaluation in Section 3.3.1.

3.3.3 Agent Imitation and Goal Utility Evaluation.

The final evaluation of the agent focuses on the relative weightings of the class imitation and goal distance within the utility function. Each class performs the same traversals as in Section 3.3.2; however, each set is run with varying α used in the utility calculation. This evaluation of utility seeks to determine the point at which the agent's ability to reach the waypoint decreases as a result of trying to obtain a better class representation. This is executed by dictating a range of α values from $\alpha = 1$ to $\alpha = 0$ at 0.1 increments for the utility calculation in Equation 7. The resulting agent's imitation evaluation and goal completion success at each value of α shows the relationship between goal utility and imitation utility.

IV. Results

The evaluations of our agent and its component pieces, as described in Section 3.3, are outlined here. We begin with the evaluations of the classification mechanism utilized in Section 3.2.3.4 and detailed in Appendix A. Next, the agent's ability to imitate a chosen VBS2 Readiness Posture is evaluated through classification accuracy in Section 4.2. Finally, the effects of the utility balance between goal achievement and class accuracy is examined.

4.1 Classifier Performance Evaluation

All five VBS2 Readiness Postures were logged as they traversed the area of interest 2000 times in 60 distinct paths. These logs are evaluated through the feature generation and classification mechanisms as described in Section 3.3. After a leave one out validation of the feature data, the resulting accuracy results are pooled in an equal weighted accuracy for each class. These results are shown in Table 3.

As seen in the confusion matrix, the classifier is able to classify each withheld feature sample with an accuracy no less than 0.141 and no greater than 0.42 for any class. Aside from the "Safe" class, all of the other classes have a classification accuracy greater than 0.305. These results indicate that the classifier is able to

Table 3. Classifier Accuracy where rows represent the generated classes and columns represent the average prediction rates of each class across a leave one out cross validation

	Aware	Careless	Combat	Safe	Stealth
Aware	0.352	0.107	0.174	0.083	0.286
Careless	0.249	0.305	0.077	0.075	0.296
Combat	0.050	0.029	0.420	0.169	0.333
Safe	0.220	0.052	0.217	0.141	0.371
Stealth	0.225	0.045	0.253	0.096	0.382

separate each class by its component feature sets with accuracies that are greater than random (except for the “Safe” class). This indicates that the classifier is capable of distinguishing between features of each class, but it also indicates that our feature generation is not exceptionally good at providing classification insight for this dataset. Similarly, These values are lower than those seen in our preliminary studies outlined in Appendix A. This is likely due to the new data set being more exhaustive and less biased toward specific feature space attributes that were easily separable between classes.

The results of the classification of the “Safe” agent’s features requires extra scrutiny. It is not surprising to see that the “Safe” behaviors are being confused with other classes, as the description of the class in Appendix B indicates that it behaves like “Aware” when in contact with enemy forces. As our scenario required the agent to traverse across a path that frequently caused the agent to come in contact with an opposing force, it is understandable that the “Safe” class would have generated feature sets that are difficult to distinguish between other classes that it behaves similarly in these kinds of situations.

It is important for the classifier to have the ability to differentiate between each class. This evaluation has shown that the classifier is able to make this type of differentiation and is a viable mechanism for the use of agent behavioral imitation.

4.2 Agent Imitation Evaluation

Similar to the evaluation of the classifier evaluation in Section 4.1 the evaluation of the agent imitation is performed by generating behaviors and classifying their resultant feature sets. The classification accuracy here indicates the agent’s ability to imitate the desired class, while incorrectly classified feature sets indicate failures of the agent’s imitated behaviors. Another factor that contributes to this evaluation is

Table 4. Agent Behavior Classified Accuracy. Constructed in a simlar way as Table 3 except the confusion matrix shows the classification accuracy of the new agent behaviors.

	Aware	Careless	Combat	Safe	Stealth
Aware	1	0	0	0	0
Careless	0	1	0	0	0
Combat	0	0	1	0	0
Safe	0	0	0	1	0
Stealth	0	0	1	0	0

the α used in Equation 7. For the purposes of this evaluation α is set to 1, effectively removing the goal location from the agent's utility calculations. Table 4 shows the agent's confusion matrix over traversals of the area of interest.

As seen in the confusion matrix the classifier classifies each agent generated sample with 100% for every class except "Stealth". The "Stealth" class is incorrectly labeled as "Combat". We believe this to be a result of our design decision in Section 3.2.3 which limited the agent to three velocities but not a stationary selection (velocity equal to zero). The rational for this decision was to prevent the agent from simply stopping actions once it was in a state that was identified as appropriate, however the adverse effect to the "Stealth" class, which travels at very slow speeds when in contact with enemies, was not predicted. This agent limitation, coupled with the class similarities indicated by the classifier results of Section 4.1 resulted in the incorrect class labeling as seen in Table 4.

Aside from this class, every other class had exemplary classification performance. This indicates that the agent is capable of producing class specific behaviors that are highly correlated with the desired class. During these class utility trials, we found that the agent behaviors were never able to achieve the goal in their 300 traversals. This also indicates that the agent behaviors, while class appropriate, do not aid in achieving a geographic objective.

It is also worth noting that these performance which are much better than those seen in the classifier accuracies of Section 4.1 are understandable within the context of how the agent uses the classifier to drive its action selection and how the feature space is divided. Since the agent has the ability to select actions based on the resulting classification of its actions, it is always able to select actions that may move it closer to the class mean in feature space. When it is time to classify the agent's performance the agent's feature vector distance from each class mean is calculated and the predicted class is the lowest distance. This mechanism has a distinct volume of success in feature space the agent need only to stay within the bounds of the class in order to succeed.

4.3 Agent Imitation and Goal Utility Evaluation

The final evaluation of the agent focuses on the relative weightings of the class imitation and goal distance within the utility function. Each class performs the same traversals as in Section 3.3.2, however, the process is repeated with α utility calculation values between 0 and 1 at 0.05 increments. This is performed at one trial per class, α, and start-goal waypoint, combination. The resulting five class average and class independent Average Imitation Correctness and Goal Accomplishment Rates across each α as shown in Figure 5.

This evaluation of utility seeks to determine the point at which the agent's ability to achieve the goal decreases as a result of systematically decreasing the weighting attributed to the goal distance utility. This tradeoff point can be seen in the Five Class Average subplot of Figure 5 at the α level of 0.6 where the goal achievement rate is approximately equal to the class accuracy. This is similarly seen in each class's independent subplots, however, at different α levels.

A higher imitation level at the lower end of α indicates that the class that is

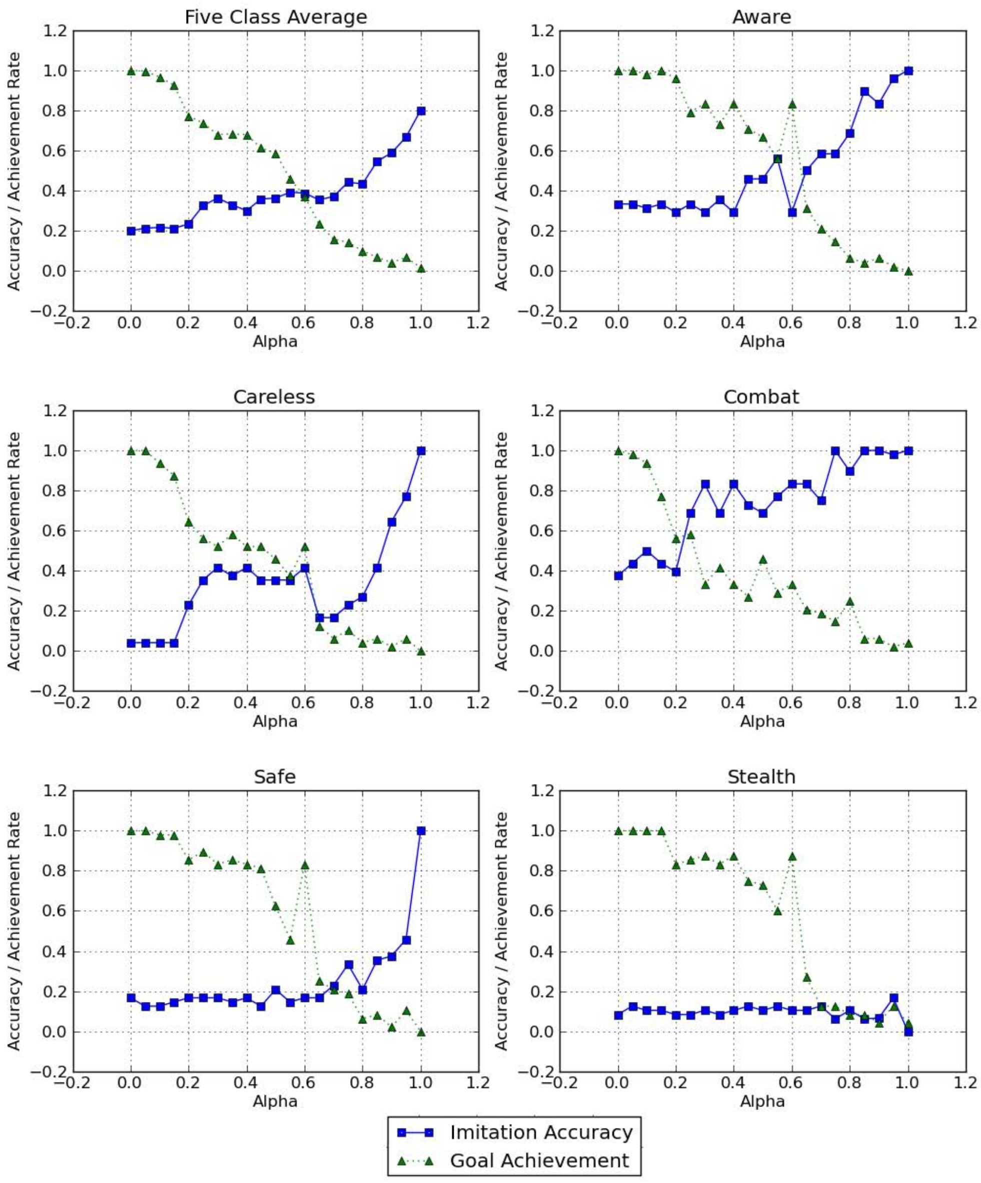


Figure 5. Average Class Imitation Accuracy and Goal Completion Rate over α

being evaluated may have been similarly goal oriented and less susceptible to altering their interactions because of the presence of the opposing forces. This is seen in the “Aware” and “Combat” classes as they have a significantly high average imitation accuracy for low values of α.

The “Stealth” class imitation accuracies are also interesting. Not only is it incapable of imitating the appropriate class for the reasons posited in Section 4.2, but it is at a level lower than what would be expected from random performance (expected to return a correct value 20% of the time). This also supports the ideas of Section 4.2 as the agent may have never been able to make actions that moved its feature vector near the “Stealth” classification center.

Overall, with the exception of the “Stealth” class, we see the expected tradeoff between class accuracy and goal utilities. This tradeoff between behavioral accuracy and goal accomplishment serves as a powerful tool that can be used to alter and vary agent performance to fit the desired training goals.

V. Conclusion

This research has shown that our behavioral imitation driven agent is capable of producing new, training class specific behaviors within the Simulated Training Environment. Given these observed behaviors, agents can be designed and implemented to utilize this additional information to imitate behaviors in tandem with other agent goals. These effective imitations can ultimately be used to improve the legitimacy and applicability of a STE by incorporating observed behaviors into the agent creation process alongside other traditional agent creation mechanisms through a utility balancing parameter α.

5.1 Analysis of Results

Our results also indicate that our agent is capable of producing class specific behaviors with a very high degree of accuracy. This ability, without the use of goal based utility, is not, however, capable of travelling to a waypoint.

We have also shown that there is a tradeoff between behavioral imitation and goal completion. This tradeoff indicates that the learned feature space representation of agent behaviors does not encapsulate enough information to drive a fully functional agent. Conversely, we have also shown that the purely goal-based utility agent is not able to imitate specific behavioral classes. Furthermore, we have shown that a mixture (α) between imitation and goal utility results in a successful agent behaviors that are both behaviorally consistent and accurate as well as able to accomplish a traversal goal.

5.2 Future Work

The data used for the training portions of our agent creation was generated by basic scripted agents. These behaviors were simple; however, they provided a difficult case study for a behavior imitation agent to replicate within this domain. This portion of the agent creation can be improved upon by using more complex agents, human traces, dynamic environments, or a combination of each. These increasingly complex data sets will yield a different feature space for the agent to adapt to.

Another addition that would be required if the complexity of the scenario were increased would be the ability to select features and weight them according to their separability. This could be accomplished by analyzing each feature with a Bhattacharya coefficient for separability between each pair of classes. The features with the greatest separability between each of the classes would be retained while those features that have low separability would be discarded.

Furthermore, not only can features be selected based on their separability, but new features can be implemented in order to encapsulate specific, measureable qualities of the agent trace. These additional features can also be evaluated and selected by the previously mentioned feature selection process.

The classification portion of our agent relies simply on the mean locations of the distributions of each class in feature space. This is a simplistic way of evaluating features because it does not take into consideration variances within the feature space of a class. Because of this neglection of variance, there are definitive lines of separation within the feature space that dictate where one class ends and another begins. This is not a representative way of evaluating new features and could be improved by taking the variance of the feature space into consideration when classifying a new sample.

The agent's ability to search through the action space is limited by the time required to take the previous action and poll the agent for a new action. In order to facilitate this capturing, analysis, and action dynamic from lagging behind the total possible move search space was condensed. For example, if the agent were quicker at evaluating a move, we could task it with evaluating all 360 degrees of possible orientations instead of the current 12. One way of improving this performance would be to multithread the process. The agent could also be tasked with a deeper search space. Currently, the agent only searches one move deep into the action space, whereas if it were multithreaded and quicker to evaluate, the agent may be able to make more appropriate choices.

The approach to balancing the agent's ability to vary the degree of influence that goal achievement and imitation accuracy through the utility calculation in Equation 7 can be improved in two ways. First, the calculated distances in feature space and physical space can be normalized to values between 0 and 1. This would prevent either of the two considerations from being discounted because of a large difference between the magnitude of the two results. Next, the utility calculation can be altered to prevent the resulting utility of either component from exponentially growing as their respective distances approaches 0. In the equation's current form, this is a result of setting the distances as the denominator. A new utility that would incorporate the desire to minimize the distance space, while preventing the exponential growth of utility as those distances approach zero can be seen in Equation 8

$$U(F_i, S'_i) = (\alpha)(1 - dF(F_i, C_j)) + (1 - \alpha)(1 - dS(S'_i, G)) \tag{8}$$

Another area of the agent design that has potential for future work is the additional, non-imitation goals that the agent tries to accomplish while maintaining a distinct behavior class. These goals can be increased in difficulty, scope, and range in order to stress and evaluate the performance of the agent's learned behaviors. Additional goals may include multiple waypoint geographic goals, hostile avoidance goals, and an adherence to a set of Rules of Engagement. These types of evaluations illustrate the versatility of this type of approach.

Similar to the addition of different goals, the principles and agents developed for this environment are easily adapted for different domains. These types of domain changes would simply require an analysis and development of features that are appropriate for the domain and the sensors available.

Appendix A. Feature Generation and Classification of Agent Behavior in a Virtual Battlespace Simulation

This appendix is the full text of a publication ready document written by 2d Lt Bryon Fryer, 2d Lt Christopher Cooper, Lt Col Brett Borghetti, and Maj Michael Mendenhall. Along with conference submission, it will also be filed as a Tech Report at the Air Force Institute of Technology.

Military organizations need realistic training scenarios to ensure mission readiness. Training the skills required to differentiate combatants from non-combatants is very important for ensuring the international law of armed conflict is upheld. In simulator-based training, one of the open challenges is to correctly simulate the appearance and behavior of combatant and non-combatant agents. One approach is to map the combatant identification problem to the more general pattern classification problem from the statistical machine learning domain. A data-driven feature selection and classifier can evaluate whether the "sweet spot" has been achieved: the point where simulated combatant and non-combatant agent behaviors are hard enough to distinguish that the training goals are met but not so hard that the two classes are indistinguishable.

In a simulated urban environment a classifier can be trained on the attributes of the agent behaviors and thus evaluate the quality of the simulated agents. With 48 derived features a Minimum Euclidean Distance classifier consistently obtained an average Equal Weighted Accuracy greater than 90% when classifying agent types in the Virtual Battle Space 2 environment.

A.1 Introduction

Military operations occur increasingly in urban, civilian-occupied areas, and the need for the ability to rapidly distinguish between combatants and non-combatants in areas of interest is clear. Military organizations invest heavily in training their members to handle difficult hostile situations. While there is no substitute for a live training exercise, virtual training through simulation is playing an ever-increasing role in military training. One of the issues in simulation-based training is the believability of the computer-generated (CG) agents in the environment. For a training simulation to be effective, the CG agents must be representative of their live counterparts not just in appearance, but also in behavior. Simulated combatants should take actions appropriate for opposing forces, while simulated non-combatants should behave according to the demographic they represent in the situation.

The overarching goals of research in this area are:

1. Improve the realism of CG agents by providing believable agents.
2. Develop methods for objectively measuring believability of CG agents.

This research focuses on the second goal - developing methods for measuring believability of computer-generated agents. Our approach is to map the problem of determining believability to the problem of pattern classification [16]. Much of the work in pattern classification can be categorized into three areas:

- Finding a set of features that accurately and efficiently represents the entities which need to be classified
- Determining to which of several classes an entity belongs
- Determining how well a certain classifier performs in the role of classifying a set of entities

In distance-based classifiers, the distance (in feature-space) from an entity to it's true class median or mean could be used to characterize the believability that the entity is really in that class: the further the entity is from it's class, the less believable it will be that it is part of the class. In the simulated agent domain, given a distance-based classifier and well-chosen features, the distance an agent is from it's true class is a measure of how believable that agent will be in the simulation.

We must stress the importance of having a good classifier and features. Without these, we would not be able to justify the statement that classification distance implies believability. This paper presents a feature-selection method and a distance-based classifier that will perform well in classifying agents in the domain. Our classification experiment is conducted with CG agents operating in the Virtual Battle Space 2 (VBS2) training environment. The experiment uses various behavior types for scripted agents traveling to a specified destination. From the agents' location and orientation data gathered in the experiment, we generated representative features to classify the behaviors of the scripted agents. A Minimum Euclidean Distance (MED) classifier trained on a the data is able to classify a novel agent's behavior into the closest class. The independent separability of each feature is measured using the Bhattacharyya Coefficient (BC) [4] between classes. Equal Weighted Accuracy (EWA) is the measurement used to characterize the quality of the classifier at predicting the true class of the agents from their behavior.

The remainder of this paper is organized as follows: In the next section, we discuss related work in the area. Section A.3 identifies the research problem and presents the four phases of our approach. In Section A.4 the feature selection and classification results are discussed. Finally, we conclude and present our thoughts on future work in the area in Section A.5.

A.2 Related Work

Within the area of agent behavior classification, much work involves the world of RoboCup Soccer. Feature extraction and classification is used to recognize and predict the behavior of robots in the opposing team. Riley and Veloso implement a time windowing technique to extract simple features and classify the current adversary into one of several predefined classes for the purpose of robotic adaptation within the RoboCup domain [26]. Although applied to a different domain, this process remains very similar to the one presented in this paper.

Wendler and Bach employ case-based reasoning as a means to classify and predict the behavior of agents in RoboCup [39]. They use simulations of games to train the classifier to recognize behavior patterns and assume that similar triggering situations lead to similar behavior patterns. Steffens [32] focuses on tactical moves of the opponent and applies a feature-based opponent modeling framework on observed data. He matches this data to the most appropriate model and selects a counter-strategy.

Clustering techniques often support classification work. Wunstel's team applied self-organizing maps on RoboCup player data to cluster typical agent behavior patterns in order to detect characteristic features of trajectories [40]. Francois' group uses self-organizing maps and classification for on-line behavior adaption [17]. Their work includes a proof-of-concept for adaptive human-robot interaction scenarios, in particular focusing on autism therapy.

A.3 Research Problem & Approach

Our research goal is to derive and evaluate features of scripted agent behavior and discover if a classifier is capable of distinguishing between the behavioral attributes of each type of agent. The work was accomplished in four main phases. The first phase (Agent Simulation Setup) focused on building the simulation environment, designing scenarios and tailoring the data extraction methods. The second phase (Path-based Classification) explored feature extraction, generation, and classifiers to determine the separability of two different agent paths in the scenario. The third phase (Behavior-based Classification) introduced five agent behavior types in a modified scenario, explored rank ordering of features for classification quality, and analyzed classifier performance in both pair-wise classification of the agents by observations of their behavior. The final phase (Limited Observability Behavior Classification) tests the the classification mechanism by performing a multi-class classification when agent behavior is only observable for a short duration instead of the full lifetime of the CG agent.

A.3.1 Phase 1: Agent Simulation Setup.

One of the first tasks was the determination of the simulation and the virtual environment. Virtual Battlespace 2 Virtual Training Kit (VBS2 VTK), an out-of-the-box simulated battlespace training solution used by the US Military for realistic and immersive training scenarios, stood out as the most applicable simulation environment. It is also used for developing and visualizing combat events, search and rescue operations and humanitarian efforts unfolding in the simulated domain. Our virtual environment was a recreation of the Ohio State University campus. We chose this map because it features an urban environment and approximately flat terrain. Agents moving in this environment will be affected by the buildings and roads.

In order to retrieve data from simulated scenarios we used a VBS2 data logger: a VBS2 plug-in that interfaces with the simulation environment and records desired simulation attributes to a log file. For our experiment, the data logger captured the location (East, North, Up), orientation (degrees), and agent ID of all agents within the simulation at a sampling rate of 1 Hz.

In this phase we created a single scenario and two agents with different behavior types. The scenario had a fixed start point and destination for each agent. The two agents differed in the way they traveled. Agent 1 used VBS2's built-in pathfinding algorithm to find a path from start to destination. Agent 2 was given specific waypoints that it had to take as it traveled from start to destination (using VBS2's pathfinding for inter-waypoint travel). Other than the fact that one type of agent was given waypoints and the other was not, both agents were the same: they both were civilian agents with VBS2 default behavior and skill levels. In addition to the buildings present on the Ohio State campus, a congregation of loitering civilians was placed near the scenario's destination point to force interaction between the agents and other mobile entities in the environment.

Because the waypoints force one agent to have a different path from the one chosen by the non-waypoint agent, the paths of the two scripted agents clearly differentiates the agents to an outside observer. In Figure A.1 the paths of the agents are shown in blue (waypoint agent) and red (non-waypoint agent). In this figure, the the civilian population is represented by the green spot in the northwest corner of the scenario. The straight line from the start location to the finish location indicates the 'as-the-crow-flies' path from the start to the destination.

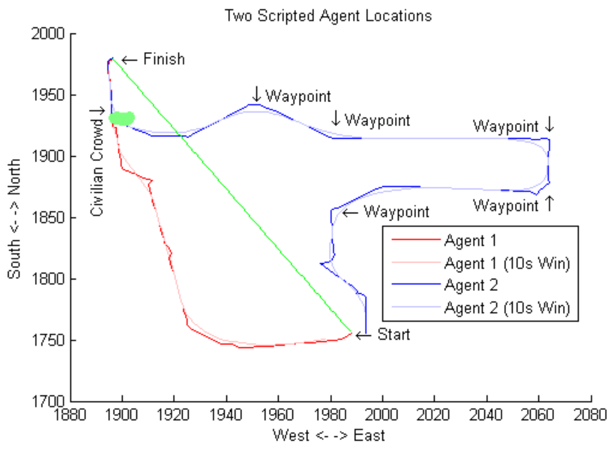


Figure A.1. Two Scripted Agent Trails. Graph axes indicate relative position in meters. Agent 1 is given a start and destination point and uses built-in pathfinding to travel to the objective. Agent 2 is given fixed waypoints which must be traversed on the route to the destination. Paths and smoothed averages (10 second windows) are shown for each agent. The 'as-the-crow-flies' direct route between start and finish is also shown.

A.3.2 Phase 2: Path-based Classification.

This phase encompassed two tasks using the data collected in the first phase: feature generation and classification. Using the data logged from the scenario in phase one we were able to select features and train a classifier which could identify which agent type (waypoint or non-waypoint) was operating in the scenario.

Feature generation dominated the workload in this phase. From the simulation environment, VBS logged the position and orientation of the agents at a frequency of 1 Hz. In order to use the data gathered for classification purposes, we collected all of the position and orientation data for the entire path each agent took, then reduced the representation of each agent's path to a single representative point in multidimensional space: one point represents the entire path the agent took in the scenario. To capture the essential information about the behavior of each agent, we generated 48 representative features[1] for each trail. These features included average location, deviation from the 'as the crow flies' straight-line path (shown in Figure A.1), directional movements quantized as shown in Figure A.2, and various other derived calculations from the measured information about the trail.

The feature generation step provided samples to train a classifier. We used a k-nearest neighbor classifier with k equal to one and determined the distance to neighbors by the Euclidean distance between a sample's 48-dimensional location and each class's mean and median location. With 30 samples of each agent performing the directed task, we trained the classifier on 27 of the samples from each class and tested on each of the 3 left out samples. We used K-Fold Cross Validation with K=10

[1]Features were **1**: goal completion time; **2**: Average East position in meters (E); **3**: Average North position in meters (N); **4**: Average Up position in meters (U); **5**: Orientation in degrees (O); **6-9**: variance on E, N, U, O; **10-13**: total change in E, N, U, O; **14-21**: total move count in each of eight cardinal directions (N, NE, E, . . . , SW); **22**: total O change count while stationary; **23**: total standing still count; **24-27**: average change in E, N, U, O; **28**: total moves made; **29**: total deviation (from 'as the crow flies' shortest possible path); **30**: average deviation; **31-38**: $\frac{\text{features } 14-21}{\text{feature } 28}$; **39-48**: $\frac{\text{features } 14-23}{\text{sample count}}$.

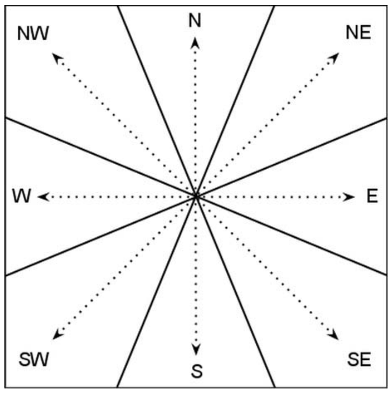


Figure A.2. Relative Agent Movement Directions. Agent movement is quantized into one of eight key directions.

(90% training data, 10% testing data) to ensure the classifier didn't just get lucky on one specific choice of testing or training data. We used Equal Weighted Accuracy (EWA) to characterize classification performance.

For each class, the classifier correctly identified each test sample 100% of the time for both the mean as well as median Euclidean distance classification. This proof-of-concept experiment successfully demonstrated that a MED classifier can accurately differentiate between paths of two scripted agents in this simulated domain. It also implied the feasibility of a behavioral classifier for classifying agents in the problem domain if the agents behavior is revealed by the movements they make when trying to achieve an objective.

A.3.3 Phase 3: Behavior-based Classification.

With the path-based classification method showing promise in the two-class case, we proceeded to further refine the system by evaluating classification of multiple classes of agents with subtler differences between the agents' behaviors. VBS2 provides a set of combat behavior attribute settings which are described in detail in the VBS2 VTK Manual [6]. These attributes define the manner in which an agent reacts to its environment and include the following types: Careless, Safe, Aware, Combat, and Stealth. Our goal was to determine the ability of the classifier to differentiate these five agent types based on their behavior within the simulated environment.

For this phase each agent was given the same starting point and destination, but no waypoints were provided for the agents. The only differences between agent types was the setting of their combat behavior attribute.

To test our classifier, we needed to redesign the scenario so that there was something in the environment for the agents to react to. In order to trigger the reactive combat behaviors unique to each agent type, we replaced the civilian crowd with two small groupings of forces on opposite sides of a conflict. Each agent being classified was made a member of one of the combat forces. Based on observations in the previous phase, we determined that the variance between simulation runs within the same scenario were relatively small and we chose to simulate ten runs of each agent behavior type for a total of fifty samples.

With the new data, we extended the feature generation and classification methods described in phase one. Since each feature used in classification adds computational complexity to the classification process, we sought to determine which features for this data-set were most beneficial for the classification algorithm. Our goal was to obtain the same classification performance faster by elimination non-contributing features. The Bhattacharyya Coefficient (BC) provides a means to rank order the features by their separability. We calculated the BC of each feature between all pairs of classes and evaluated the average of these coefficients for each feature in order to determine the features that provided the most (relative) separability across all of the data collected. We ran a pair-wise classification between each of the five behavior types, leaving one sample out from each of the two selected classes (retaining a 90% training data and 10% testing data ratio).

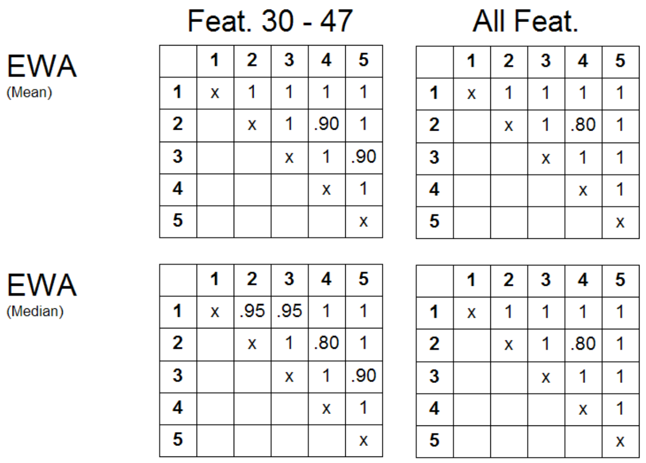
A.3.4 Phase 4: Limited Observability Behavior Classification.

While achieving strong classification results in the multi-class task is necessary, it is unlikely that in the real world a trainee would have visibility of the entire lifetime of any one CG agent. It is much more likely that the trainee would be able to observe the CG agent for a relatively short period of time during it's lifespan. Given this limitation, we decided to explore what would happen if a classifier had to make a classification decision under such circumstances.

In order to test the performance of the classifier when it had a limited observation of a CG agent's behavior, we divided the agent trails into smaller time windows. We varied the window length from 2 seconds through 80 seconds. Smaller window lengths decrease the number of different values a feature could take on, while simultaneously increasing the number of samples available for training and testing. We ran a full feature generation, classification and accuracy determination using ten-fold cross validation. For each window length, we calculated the all-class EWA of the classifier (in contrast to all-possible-pairwise-comparison EWA evaluated in phase three).

A.4 Results

In phase three, average EWA (mean MED and median MED) was calculated using all 48 derived features for ten iterations of K-fold cross validation (where k=10) of each pairwise comparison between the five classes. Average EWA was also calculated on a subset of eighteen features chosen because they represent the normalized number of directional moves and orientation changes of the agents. Results of the EWA for both the subset and the full set of features are depicted using a EWA matrix in Figure A.3. In pairwise classification, the EWA of the full feature set outperforms the EWA of the classifier trained using a smaller set of features in all but one case: Class 2 (Careless) vs Class 4 (Safe).



EWA (Mean)

Feat. 30 - 47

	1	2	3	4	5
1	x	1	1	1	1
2		x	1	.90	1
3			x	1	.90
4				x	1
5					x

All Feat.

	1	2	3	4	5
1	x	1	1	1	1
2		x	1	.80	1
3			x	1	1
4				x	1
5					x

EWA (Median)

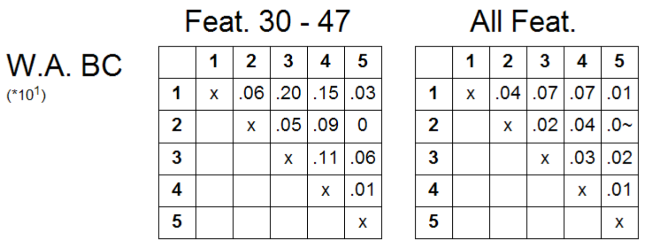
Feat. 30 - 47

	1	2	3	4	5
1	x	.95	.95	1	1
2		x	1	.80	1
3			x	1	.90
4				x	1
5					x

All Feat.

	1	2	3	4	5
1	x	1	1	1	1
2		x	1	.80	1
3			x	1	1
4				x	1
5					x

Figure A.3. Pairwise Average EWA. The value in a cell indicates a binary classifier's ability to separate the class indicated in the row from the class indicated in the column.



W.A. BC ($*10^1$)

Feat. 30 - 47

	1	2	3	4	5
1	x	.06	.20	.15	.03
2		x	.05	.09	0
3			x	.11	.06
4				x	.01
5					x

All Feat.

	1	2	3	4	5
1	x	.04	.07	.07	.01
2		x	.02	.04	.0~
3			x	.03	.02
4				x	.01
5					x

Figure A.4. Weighted Average BC. The value in a cell indicates the average Bhattacharyya Coefficient for all considered features when attempting to differentiate the row class from the column class. The closer the number to zero, the easier the two classes will be to separate using those features.

We then estimated which of the class pairings are more separable than others using modified Bhattacharyya Coefficients as a heuristic. In order to accomplish this we determined the average BC between each pairwise set of classes across both the reduced feature set and all features. These values were then weighted by the number of features taken into consideration. The result was the Weighted Average Bhattacharyya Coefficients (W.A. BC) for each pairwise class comparison and for each set of features. These results can be seen in Figure A.4. A higher W.A. BC value implies that a classifier using those features would have a harder time separating the two classes.

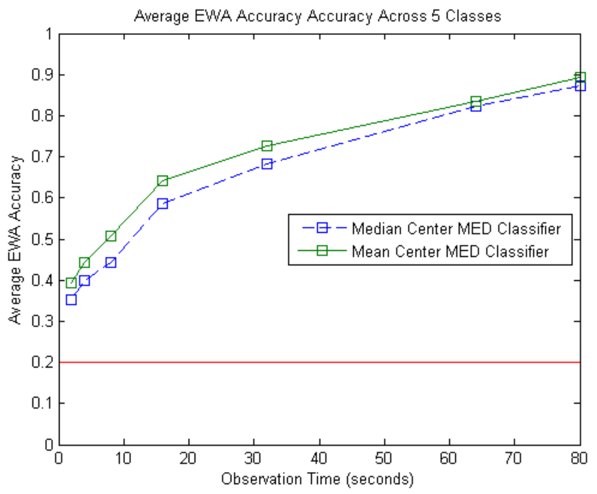


Figure A.5. Five-class EWA as a function of observed time window length. The horizontal line at 0.2 indicates the accuracy of a purely random selection.

The goal of phase three was to predict which class (from the full set of five classes) a CG agent was from. The classifier was capable of correctly identifying a sample with an average EWA of 96% and 98% using the median and mean respectively as the measures of center. The only mis-classifications occurred between the "Safe" and "Careless" classes which are described in the VBS2 VTK Manual as being very similar in appearance [6].

In phase four we explored how the length of time an agent was able to be observed affected classification accuracy. We found that using the smallest time window (2 seconds of observed behavior) our classifier was able to correctly identify an agent twice as often as would be predicted by a random choice, indicating that these feature selection and classification methods work even in conditions of limited time visibility. In the region we tested (2 to 80 seconds), classification performance increases at a rate proportional to the log of the window size. These accuracies are shown in Figure A.5.

A.5 Conclusion and Future Work

Our research showed that classification of features derived from the position and orientation of agents is a potential mechanism for identifying an agent based on their behavior in the simulation. The MED classifier was successful in identifying scripted agents by their observed behaviors based on the slight differences in their derived features, even when the features were derived during very short windows of visibility. Given the assumption that humans working in simulated training environments are looking at the same types of features when deciding how believable a CG agent is, this kind of classification mechanism can be used to measure believability of the agents.

These results are contingent on assumptions about a human's characterization of believability as well as experiment design assumptions and simulation environment limitations. Further exploration of the feature selection and classification is necessary in several areas: 1) What are the features that a human observer actually bases a decision on when determining the believability of an agent? 2) How can we do better at selecting a subset of features that would maximize classifier performance? 3) What other classifiers might be more appropriate than the MED classifier?

Acknowledgments

The authors would like to thank the Air Force Research Laboratory's 711th Human Performance Wing for their sponsorship of this research and Numerica Corporation for their technical assistance.

The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government

Appendix B. VBS2 Specifications and Definitions

B.1 Combat Modes

This setting defines whether or not the AI group will engage enemy targets. The available options are:

- **No change**: The group will continue under its existing combat mode.
- **Never fire**: The leader will order the group to hold fire and disengage. The group will not fire under any circumstance. (Also known as combat mode BLUE)
- **Hold fire**: The leader will order the group to hold fire and disengage. Individual units may open fire on any enemy units that are a threat. (Also known as combat mode GREEN)
- **Hold fire, engage at will**: The leader will order the group to hold fire but engage enemy units at will. Individual units will move into a position from where they may shoot at the enemy, but will only open fire on an enemy unit that becomes a threat. (Also known as combat mode WHITE).
- **Open fire**: This is the default Combat Mode. Units will fire upon any suitable target in range, while staying in formation. The group leader may order individual units to engage targets. (Also known as combat mode YELLOW)
- **Open Fire, Engage At Will**: The leader will order units to fire upon any suitable target in range, and move to engage at will. Units may move out of formation in order to find suitable firing positions on known targets. (Also known as combat mode RED)

B.2 Readiness Postures (Behaviors)

Behaviors define the manner in which a group moves from one point to another. Combat mode overrides all other movement setting (ie Combat Mode, Formation and Speed). Available settings are:

- **No change**: The group behavior will remain as current.
- **Careless**: Careless behavior will cause the group to move and behave in a very non-combat manner. The group will form into a Compact Column like formation, where each unit will directly follow the man in front rather than the group leader. Soldiers will carry their weapons in safe position (rifles across body, pistols holstered) and walk slowly. Infantry will not fire on enemy targets (unless they are shot at), but vehicles will still fire on enemies when encountered. Groups in careless mode do not switch to a more alert mode if enemies are encountered. All units show preference moving along roads whenever possible.
- **Safe**: Similar to Careless, except the group will change behavior to Aware upon detecting an enemy unit.
- **Aware**: This is the default behavior mode. The group will move at moderate speed, with soldiers generally standing upright and making some occasional efforts to use cover when available. Most units will still prefer to travel along roads and travel in convoy irrespective of formation type. Tracked vehicles will not use headlights, and will drive across any surface with no preference to staying on roads. Helicopters will not use searchlights. When enemies are known to be in the area, troops will disembark from any of their group's wheeled transport vehicles (trucks, cars), and the group will move while carrying out "bounding" maneuvers, making stronger use of available cover.

- **Combat**: This behavior mode will result in a much higher combat performance than Aware. Infantry groups will always move using bounding maneuvers, and will normally keep crouched or prone unless moving. They will make some use of available cover, choosing to spend some time crawling when in cover. They will occasionally send out one unit ahead of the group as a scout. No vehicles will use headlights at night. If enemy units are known to be in the area, infantry groups will move in a more cautious manner.
- **Stealth**: Stealth mode will cause a group to behave in a more cautious manner. Infantry groups will move via cover whenever possible, spending much of their time crawling. When they need to cross open ground, they appear to occasionally choose to send scouts running ahead to reach the cover ahead as quickly as possible. A stealthy infantry formation can tend to end up quite fractured. Wheeled vehicles will still follow roads if available, but no longer convoy. If enemy units are known to be in the area, infantry groups will move more closely together and spend more time prone.

B.3 Unit Stance

Unit Stance defines the starting posture of the AI unit. This setting will not be overridden by the group commanders current waypoint. Available settings are:

- **AI controlled**: The units posture will be controlled by the AI
- **Standing**: The unit will always remain standing
- **Kneeling**: The unit will kneel, unless moving to point to point where the unit will stand
- **Prone**: The unit will always remain lying, even when moving from point to point where the unit will crawl

Appendix C. Average Class Imitation Accuracy and Goal Completion Rate over α

α	goal	cls	c0ga	c1ga	c2ga	c3ga	c4ga	c0ia	c1ia	c2ia	c3ia	c4ia
0.00	1.00	0.20	1.00	1.00	1.00	1.00	1.00	0.33	0.04	0.38	0.17	0.08
0.05	1.00	0.21	1.00	1.00	0.98	1.00	1.00	0.33	0.04	0.44	0.13	0.13
0.10	0.97	0.22	0.98	0.94	0.94	0.98	1.00	0.31	0.04	0.50	0.13	0.10
0.15	0.93	0.21	1.00	0.88	0.77	0.98	1.00	0.33	0.04	0.44	0.15	0.10
0.20	0.77	0.23	0.96	0.65	0.56	0.85	0.83	0.29	0.23	0.40	0.17	0.08
0.25	0.74	0.33	0.79	0.56	0.58	0.90	0.85	0.33	0.35	0.69	0.17	0.08
0.30	0.68	0.36	0.83	0.52	0.33	0.83	0.88	0.29	0.42	0.83	0.17	0.10
0.35	0.68	0.33	0.73	0.58	0.42	0.85	0.83	0.35	0.38	0.69	0.15	0.08
0.40	0.68	0.30	0.83	0.52	0.33	0.83	0.88	0.29	0.42	0.83	0.17	0.10
0.45	0.61	0.36	0.71	0.52	0.27	0.81	0.75	0.46	0.35	0.73	0.13	0.13
0.50	0.59	0.36	0.67	0.46	0.46	0.63	0.73	0.46	0.35	0.69	0.21	0.10
0.55	0.46	0.39	0.56	0.38	0.29	0.46	0.60	0.56	0.35	0.77	0.15	0.13
0.60	0.37	0.39	0.83	0.52	0.33	0.83	0.88	0.29	0.42	0.83	0.17	0.10
0.65	0.23	0.35	0.31	0.13	0.21	0.25	0.27	0.50	0.17	0.83	0.17	0.10
0.70	0.16	0.37	0.21	0.06	0.19	0.21	0.13	0.58	0.17	0.75	0.23	0.13
0.75	0.14	0.44	0.15	0.10	0.15	0.19	0.13	0.58	0.23	1.00	0.33	0.06
0.80	0.10	0.43	0.06	0.04	0.25	0.06	0.08	0.69	0.27	0.90	0.21	0.10
0.85	0.07	0.55	0.04	0.06	0.06	0.08	0.08	0.90	0.42	1.00	0.35	0.06
0.90	0.04	0.59	0.06	0.02	0.06	0.02	0.04	0.83	0.65	1.00	0.38	0.07
0.95	0.07	0.67	0.02	0.06	0.02	0.10	0.13	0.96	0.77	0.98	0.46	0.17
1.00	0.02	0.80	0.00	0.00	0.04	0.00	0.04	1.00	1.00	1.00	1.00	0.00

Appendix D. Scenerio Generation Specifics

Table D.1. X Cordinates

x_0	325
x_1	375
x_2	425
x_3	475

Table D.2. Y Cordinates

y_0	2100
y_1	2150
y_2	2200
y_3	2250

Table D.3. Start Locations

s_0	x_0, y_0	325, 2100
s_1	x_0, y_1	325, 2150
s_2	x_0, y_2	325, 2200
s_3	x_0, y_3	325, 2250
s_4	x_1, y_3	375, 2250
s_5	x_2, y_3	425, 2250
s_6	x_3, y_3	475, 2250
s_7	x_3, y_2	475, 2200
s_8	x_3, y_1	475, 2150
s_9	x_3, y_0	475, 2100
s_{10}	x_2, y_0	425, 2100
s_{11}	x_1, y_0	375, 2100

Table D.4. Traversal Selections

Start Location	Destinations
s_0	s_4, s_5, s_6, s_7, s_8
s_1	s_5, s_6, s_7, s_8, s_9
s_2	s_6, s_7, s_8, s_9, s_{10}
s_3	s_7, s_8, s_9, s_{10}, s_{11}
s_4	s_8, s_9, s_{10}, s_{11}, s_0
s_5	s_9, s_{10}, s_{11}, s_0, s_1
s_6	s_{10}, s_{11}, s_0, s_1, s_2
s_7	s_{11}, s_0, s_1, s_2, s_3
s_8	s_0, s_1, s_2, s_3, s_4
s_9	s_1, s_2, s_3, s_4, s_5
s_{10}	s_2, s_3, s_4, s_5, s_6
s_{11}	s_3, s_4, s_5, s_6, s_7

Bibliography

[1] Aler, R., J.M. Valls, D. Camacho, and A. Lopez. "Programming Robosoccer agents by modeling human behavior". *Expert Systems with Applications*, 36(2):1850–1859, 2009.

[2] Bain, M. and C. Sammut. "A Framework for Behavioural Cloning". *Machine Intelligence 15*, 103–129. Oxford University Press, 1996.

[3] Bakker, Paul and Yasuo Kuniyoshi. "Robot See, Robot Do : An Overview of Robot Imitation". *In AISB96 Workshop on Learning in Robots and Animals*, 3–11. 1996.

[4] Bhattacharyya, A. "On a measure of divergence between two statistical populations defined by probability distributions". *Bull. Calcutta Math. Soc*, 35:99–109, 1943.

[5] Blascovich, J., J. Loomis, A.C. Beall, K.R. Swinth, C.L. Hoyt, and J.N. Bailenson. "Immersive Virtual Environment Technology as a Methodological Tool for Social Psychology". *Psychological Inquiry*, 13(2):103–124, 2002.

[6] Bohemia, Interactive. "VBS2 Editor Manual 1.02 : Offline Mission Editor Real-Time Editor". Website, 2007. Bohemia Interactive Australia Pty. Ltd.

[7] Brockington, M. and M. Darrah. "How not to implement a basic scripting language". *AI Game Programming Wisdom*, 548–554, 2002.

[8] Brown, J.S. and R.R. Burton. "Diagnostic models for procedural bugs in basic mathematical skills". *Cognitive science*, 2(2):155–192, 1978.

[9] Buro, M. and T. Furtak. "RTS Games as Test–Bed for Real–Time AI Research". *Proceedings of the 7th Joint Conference on Information Science (JCIS 2003)*, 2003.

[10] Dahlbom, A. *An adaptive AI for real-time strategy games.* Ph.D. thesis, University of Skovde, School of Humanities and Informatics, 2004.

[11] Dahlbom, A. and L. Niklasson. "Goal-directed hierarchical dynamic scripting for RTS games". *Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference*, 21–28. 2006.

[12] Davis, J. "OSU GIS data". Dept. of Computer Science & Engineering, Ohio State University. www.cse.ohio-state.edu/$\tilde{j}$wdavis.

[13] Devijver, P. and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall, 1982.

[14] DHS. "Secret Service Site Security Training Gains a High-Tech Edge". Website, Jan 2011. www.dhs.gov/files/programs/gc_1295637658955.shtm.

[15] Dompke, U. "Computer Generated Forces-Background, Definition and Basic Technologies". *Simulation of and for Military Decision Making*, 2003.

[16] Duda, Richard O., Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2 edition, November 2001. ISBN 0471056693. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0471056693`.

[17] Franccois, Dorothée, Daniel Polani, and Kerstin Dautenhahn. "On-line behaviour classification and adaptation to human-robot interaction styles". *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, HRI '07, 295–302. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-617-2. URL `http://doi.acm.org/10.1145/1228716.1228756;http://doi.acm.org/10.1145/1228716.1228756`.

[18] Guadagno, R.E., J. Blascovich, J.N. Bailenson, and C. Mccall. "Virtual humans and persuasion: The effects of agency and behavioral realism". *Media Psychology*, 10(1):1–22, 2007.

[19] Hoffman, D. "Analysis of a Taxonomy for Test Oracles". *Quality Week*, volume 98. 1998.

[20] Interactive, Bohemia. "Bohemia Interactive announce USMC Partnership". Website, 2009. http://www.bisimulations.com/index.php?&Itemid=73.

[21] Laird, J.E. "An Exploration into Computer Games and Computer Generated Forces". *Eighth Conference on Computer Generated Forces and Behavior Representation*, 2000.

[22] Maybury, Mark T. "Technology Horizons : A Vision for Air Force Science & Technology During 2010-2030". *Office of the Chief Scientist of the U.S. Air Force*, 2010.

[23] Nakano, A., A. Tanaka, and J. Hoshino. "Imitating the Behavior of Human Players in Action Games". *Entertainment computing–ICEC 2006: 5th international conference, Cambridge, UK, September 20-22, 2006: proceedings*, 332. Springer-Verlag New York Inc, 2006. ISBN 3540452591.

[24] Numerica, Corp. "SimExec Agent". Digital Software, 2010.

[25] Numerica, Corp. "VBS2 Logger Plugin". Digital Software, 2010.

[26] Riley, P. and M. Veloso. "On Behavior Classification in Adversarial Environments". *Distributed Autonomous Robotic Systems 4*, 371–380. Springer-Verlag, 2000.

[27] Riley, P. and M. Veloso. "Coaching a simulated soccer team by opponent model recognition". *Proceedings of the fifth international conference on Autonomous agents*, 155–156. ACM, 2001.

[28] Robson, S. "Army paying 17.7M for training game". Website, Jan 2009. http://www.stripes.com/news/army-paying-17-7m-for-training-game-1.86770.

[29] Scott, B. "The illusion of intelligence". *AI Game Programming Wisdom*, 16–20, 2002.

[30] Spronck, P., M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma. "Adaptive game AI with dynamic scripting". *Machine Learning*, 63(3):217–248, 2006.

[31] Spronck, P., I. Sprinkhuizen-Kuyper, and E. Postma. "Online adaptation of game opponent AI in simulation and in practice". *Proceedings of the 4th International Confrence on Intelligent Games and Simulation (Game-ON 2003)*, 93–100, 2003.

[32] Steffens, Timo. "Feature-based declarative opponent-modelling in multi-agent systems". *Publications of the Institute of Cognitive Science*, 4-2004(Septemeber), 2004.

[33] Thurau, C., G. Sagere, and C. Bauckhage. "Imitation Learning at All Levels of Game AI". *Proceedings of the international conference on computer games, artificial intelligence, design and educatoin*, 2003.

[34] Tozour, P. "The evolution of game AI". *AI Game Programming Wisdom*, 3–15, 2002.

[35] Tozour, P. "The perils of AI scripting". *AI Game Programming Wisdom*, 541–547, 2002.

[36] Valiant, L.G. "A theory of the learnable". *Communications of the ACM*, 27(11):1134–1142, 1984. ISSN 0001-0782.

[37] Webb, G.I., M.J. Pazzani, and D. Billsus. "Machine learning for user modeling". *User Modeling and User-Adapted Interaction*, 11(1):19–29, 2001.

[38] Whetzel, J.H. *Developing intelligent agents for training systems that learn their strategies from expert players.* Ph.D. thesis, Texas A&M University, 2005.

[39] Wndler, Jan and Joscha Bach. *Recognizing and Predicting Agent Behavior with Case Based Reasoning*, volume 3020 of *RoboCup 2003: Robot Soccer World Cup VII*, 729–738. Springer, Berlin, 2004.

[40] Wunstel, Michael, Daniel Polani, Thomas Uthmann, and Jurgen Perl. "Behavior Classification with Self-Organizing Maps". *RoboCup 2000: Robot Soccer World Cup IV*, 108–118. Springer-Verlag, London, UK, 2001. ISBN 3-540-42185-8. URL `http://portal.acm.org/citation.cfm?id=646585.698842`.

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 074-0188*
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD-MM-YYYY)* 24-03-2011	**2. REPORT TYPE** **Master's Thesis**		**3. DATES COVERED** *(From – To)* Aug 2009- Mar 2011
4. TITLE AND SUBTITLE VIRTUAL BATTLESPACE BEHAVIOR GENERATION THROUGH CLASS IMITATION		**5a. CONTRACT NUMBER**	
		5b. GRANT NUMBER	
		5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Bryon K. Fryer Jr., 2d Lt, USAF		**5d. PROJECT NUMBER**	
		5e. TASK NUMBER	
		5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765			**8. PERFORMING ORGANIZATION REPORT NUMBER** ~~AFIT/GE/ENG/11-12~~ AFIT/GCO/ENG/11-04
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. John Duselis Program Manager - Anticipate and Inuence Behavior Division John.Duselis@wpafb.af.mil – 937–255–3219 Air Force Research Labs, 711th Human Performance Wing Bldg 248 2255 H Street WPAFB, OH 45433-7022			**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/711 HPW/RHXB
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.			
14. ABSTRACT Military organizations need realistic training scenarios to ensure mission readiness. Developing the skills required to differentiate combatants from non-combatants is very important for ensuring the international law of armed conflict is upheld. In Simulated Training Environments, one of the open challenges is to correctly simulate the appearance and behavior of combatant and non-combatant agents in a realistic manner. This thesis outlines the construction of a data driven agent that is capable of imitating the behaviors of the Virtual BattleSpace 2 behavior classes while our agent is configured to advance to a geographically specific goal. The approach and the resulting agent promotes and motivates the idea that Opponent and Non-Combatant behaviors inside of simulated environments can be improved through the use of behavioral imitation.			
15. SUBJECT TERMS FPGA, radiation induced faults, single event upset, total ionizing dose			

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT** UU	**18. NUMBER OF PAGES** 78	**19a. NAME OF RESPONSIBLE PERSON** Brett Borghetti, Lt Col, USAF (ENG)
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U			**19b. TELEPHONE NUMBER** *(Include area code)* (937)255-3636 x 4612 brett.borghetti@a˙t.edu

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18