Building entity models through observational learning

Richard D Garcia¹, Robert Kania², MaryAnne Fields³, and Laura Barnes⁴

Abstract
To support the missions and tasks of mixed robotic/human teams, future robotic systems will need to adapt to the dynamic behavior of both teammates and opponents. One of the basic elements of this adaptation is the ability to exploit both long- and short-term temporal data. This adaptation allows robotic systems to predict/anticipate, as well as influence, future behavior for both opponents and teammates and will afford the system the ability to adjust its own behavior in order to optimize its ability to achieve the mission goals. This work is a preliminary step in the effort to develop online entity behavior models through a combination of learning techniques and observations. As knowledge is extracted from the system through sensor and temporal feedback, agents within the multi-agent system attempt to develop and exploit a basic movement model of an opponent. For the purpose of this work, extraction and exploitation is performed through the use of a discretized two-dimensional game. The game consists of a predetermined number of sentries attempting to keep an unknown intruder agent from penetrating their territory. The sentries utilize temporal data coupled with past opponent observations to hypothesize the probable locations of the opponent and thus optimize their guarding locations.

Keywords
Mobile agent, territory guarding, invading strategies, multi-agent systems, adaptive systems

1. Introduction
Security defense is a complex problem which aims to protect sensitive regions against adversary intrusion. The use of multi-agent systems for the application of autonomous security defense is of particular relevance with the use of unmanned systems increasing on the battlefield. One of the greatest challenges is determining what the optimal move is to prevent adversaries from entering secure areas.

We consider a multi-agent defense scenario in which guard agents attempt to protect a region from intruder agents. Guard agents can protect by utilizing a fixed navigation strategy such as following a perimeter or by attempting to maximize their objective function. Amigoni et al.¹ showed that considering adversary models can maximize the expected outcome better than in those cases where the adversary model is not considered.

Variations of the guarding problem have been studied by many fields over the years. From the Museum Problem,²,³ a classic problem in computational geometry, to the Pursuit–Evasion problem, a variant of modern game theory;⁴–⁶ the concept of guarding has piqued the interest of many researchers.

The “Museum Problem”, also known as Chvátal’s Art Gallery Theorem, could be thought of as a static version of the guarding problem. Here, the objective is to determine the optimal number and placement of guards that will adequately cover a predetermined region. Variations exist with regards to placement constraints, visibility range and field-of-view constraints, and mobility constraints.

¹Southwest Research Institute, San Antonio, USA
²US Army Tank Automotive Research Development and Engineering Center, Warren, USA
³Army Research Laboratory, Aberdeen Proving Ground, USA
⁴University of South Florida, Tampa, USA

Corresponding author:
Laura Barnes, Department of Systems & Information Engineering, University of Virginia, 151 Engineer’s Way, PO Box 400747, Charlottesville, VA 22904-4747, USA.
Email: lbarnes@virginia.edu
Previous work on pursuit and evasion games has often simplified these constraints by generating strategies where the agents have complete visibility of targets within the environment. Other techniques which address imperfect knowledge are formalized to discrete domains or they pre-compute patrol strategies. While there are many shared features between pursuit evasion games and the patrolling scenario we present here, there are some differences. First, the only goal for an evader is to avoid capture—the intruder in a patrolling scenario wants to enter a specific region in the environment (e.g., a high-value target). Also, intruders are not captured and removed from the game; they are escorted from the “no-fly zone” (NFZ) to open airspace where they can plan and execute their next incursion.

A closely related area of research to this work is denial. In Subramanian and Cruz methods are examined to model pop-up targets with Markov models. In their work, the pop-up locations were random so it was not possible to utilize historical information to predict the location of the next pop-up. Also, their engagement did not consider the dynamics of the pop-up vehicle. In Vanek et al. a game theoretic approach is used to find the optimal path through a shipping lane for an intruder. In this case, the patrollers have realistic characteristics such as limited fuel and refueling bases that they must use periodically. They do not consider strategies for the patrollers, nor do they incorporate a history for the intruder into their work. In Sak et al. the problem of predictability for the guards is addressed. The researchers evaluate their modification of a Traveling Salesman Algorithm against three types of intruders—random, waiting, and statistical. One difference between their testing scenario and our scenario is that their intruders can “hide” near a target whereas ours cannot. Also, Sak et al. do not consider dynamics of the intruder or previous history.

Another related area of research involves the Capture-the-Flag (CTF) game in which teams must simultaneously defend their flag and attempt to capture an enemy’s flag. Sebastiani et al. used Bayesian clustering to classify the action sequences from CTF in an abstract setting into a small number of clusters. Hefny et al. used reinforcement learning to develop high-level behavior policies for team members. Bakkas et al. developed a team level online strategy based on the current team configuration and recent history. One difference between our work and these approaches is that we incorporate the dynamics and time history of the intruders into the online strategies for the defenders.

In this work, we explore learning based on historical observations and attempt to exploit gained information to modify basic behaviors. We consider a multi-agent pursuit scenario in which there is a team of guard agents and intruder agents. The objective of the guards is to observe the intruders’ behavior and thwart them from gaining entry into a NFZ. The intruders may move evasively, along a particular trajectory, or even incorporate some randomness. We also assume that guards have limited visibility due to characteristics such as sensor limitations, walls, obstacles, etc. The high-level goal of the guard agents is to be able to accurately predict the movement of the adversary by learning their behavior patterns. This is accomplished by monitoring the opposing forces position, movements, and actions over time. By recording recent activity and patterns in motion, behaviors of the opposing forces can be identified and utilized in creating a probabilistic map of intruder movements. This is achieved through building entity behavior models of the intruder agents.

Section 2 formalizes the problem and describes the guard and intruder behaviors. Section 3 discusses the simulation environment. Results and conclusions are presented in Sections 4 and 5, respectively.

2. Algorithms

The concept of guarding a region is relatively straightforward to define and can generally be considered a multi-turn modification or extension of the classical Pursuer–Evader problem. In this implementation, an intruder is attempting to invade a goal location placed within the patrolling area of a guard(s). The guard’s objective is to immediately intercept any intruders and escort them out of the guard’s patrolling area.

2.1 Formalized guarding problem

In this work we are attempting build an understanding of a particular intruder’s strategy by utilizing partial world knowledge and feedback that may be incomplete and time variant. We assume that it will take significant time to learn an entire model and thus focus on understanding the parts of the model that can be quickly exploited by the guards, i.e. the underlying patterns in the entity’s strategy. This problem is formally represented as $n$ guards attempting to minimize the depth and frequency of penetration into a NFZ by a group of $m$ intruder agents. Intruders and guards are assumed to have capabilities that allow them to directly view other agents within a particular range. Furthermore, it is assumed that penetration into the NFZ will immediately alert guard agents to the position of the intruder. Guard agents are confined to the NFZ at all times.

Although there is a specific location, $O$, that is considered to be the goal of the intruder, this work places limited importance on this particular location. Consequently, $O$ should not be considered a target location but should be thought of as the maximum tolerable penetration depth inside the NFZ by the intruder. In the case that $O$ is
considered a high-value target with infinite protection priority, the solution to the problem becomes deterministic. This is due to the fact that the optimal location of the guard(s) can be solved without knowledge of the intruder’s location. For example, consider the case of a single guard operating in an obstacle free circular world where \( R_N \) is the radius of the NFZ, \( R_T \) is the radius of the target located at the center of the NFZ and \( R_G \) is the distance of the guard from the center of the target. It will take an intruder \( R_N - R_T + 1 \) turns to breach the NFZ, pass through the NFZ, and penetrate the target. Assuming that both the guard and intruder move at equal speeds, it can be shown that if

\[
R_G + R_T \leq R_N - R_T
\]

that intruder will never be able to reach the target before being intercepted by the guard. Furthermore, the maximum penetration depth, \( P_I \), can be calculate using

\[
P_I = \frac{R_N + R_G}{2}
\]

Thus, in order to guarantee target protection while minimizing the possible penetration depth, the guard must minimize \( R_G \) while ensuring equation (1) remains true, i.e. \( R_G = 0 \). This leads to a static optimal guard position located at the very center of the target. Even if the guard was temporarily aware of the location of the intruder, this would only temporarily modify the optimal guarding location.

It should be noted that situations exist where equation (1) will not hold regardless of the guard’s position. In any instance where \( R_T > R_N - R_T \), no position exists where the guard can guarantee that an intruder will not reach the target area. In these situations sub-optimal solutions must be explored in order to minimize target penetration by intruders.

Although this work primarily utilizes discretized Euclidian space, the above concepts were presented using Polar space in order to better express the rationale behind this work. The concept that optimal guard locations can be calculated, if they exist, as well as the need for exploring sub-optimal solutions hold true in both Polar and Euclidian space. As this sub-section was only intended as extended justification, Polar coordinates were used as they better express the overall concepts.

This work is primarily focused on minimizing the depth and frequency of intruder penetration into the NFZ. This is done by combining known limitations about the intruder vehicle with knowledge extracted from historical encounters.

### 2.2 Guard strategy

As previously mentioned the goal of the guard is to prevent/limit entry into the NFZ. Guard strategies are built upon the assumption that both the guard and intruder can transition to any neighboring location and that both the guard and intruder agents transition at exactly the same rate (i.e. the speed of the guard and intruder are equal and constant). The only exception to this assumption is that guards are restricted to the NFZ at all times.

#### 2.2.1 Finding possible locations.

In this work, the guard utilizes two distinct inputs to calculate the optimal guarding location at time, \( t \), and the possible locations of the intruder and the past NFZ breaching locations, both in discretized Euclidian space. A possible intruder location refers to viable locations that the intruder can reside at time, \( t \) when the intruder cannot be sensed directly. Assuming that the dynamic capabilities of the intruder are known, viable locations can be determined by growing a bounded region, referred to as the possibility region, that is rooted at the last known location of the intruder. This possibility region represents the maximum range of the intruder at time \( t \) but provides no information regarding the probability that the intruder occupies any particular location.

This work assumes that an intruder’s dynamics are holonomic, or at least can be bounded using the holonomic assumption, and that its maximum speed is one distance unit per time step. In its most basic implementation, this vehicle model results in a possibility region that is centered at the intruder’s last known location and grows equally outward in all directions at each time step. This basic growth pattern is deformed by areas that can be directly sensed by the guard vehicles, i.e. locations within the NFZ or within a guard’s internal sensor range. Consequently, possible locations will be limited to growth around the NFZ.

The growth deformation caused by the sensing range of the guard(s), denoted by \( S_G \), is both time and location dependent. A location, \( L \), can be determined as a possible intruder’s location at time, \( t \), i.e. \( P(L, t) \) using

\[
P(L, t) = \begin{cases} 
1, & \text{if } (P(L, t - 1) = 1 \text{ or } N(L, t - 1) = 1) \text{ and } L \notin (NFZ \cup S_G(t)) \text{ for } \forall(t) \\
0, & \text{otherwise.}
\end{cases}
\]

where \( N(L, t - 1) \) is a function that determines if a direct neighbor of \( L \) is a viable location of an intruder at time \( t - 1 \) and \( S_G(t) \) is the set of all locations directly visible by the \( i \)th guard at time \( t \). Thus, assuming that location \( L \) is not currently within a guard’s sensor range, the intruder could be at location \( L \) if \( L \) was a viable intruder location at time \( t - 1 \) or if \( L \) is reachable from a viable intruder location at \( t - 1 \) in a single time step.

Note that equation (3) is only valid between the current time step and the time step of the last known location of the intruder. Before the initial sighting of the intruder, all
locations outside of the NFZ and guard’s sensor range are considered valid, \( P(L, t) = 1 \). From an implementation standpoint, it is computationally inefficient to calculate \( P(L, t) \) in the recursive method presented in equation (3). To increase calculation efficiency, \( P(L, t) \) can be calculated at time \( t \) and then stored for later use when calculating \( P(L, t+1) \).

Using equation (3) a discretized map can be generated where all locations on the map can be defined as possible, \( P(L, t) = 1 \), or impossible, \( P(L, t) = 0 \), locations for the intruder to currently occupy. This map, referred to as the possibility map, will be used to bound additional operations aimed at improving guards’ defensive maneuvers.

Utilizing only the possible locations of the intruder can significantly reduce calculation time and increase a guard’s potential for preventing NFZ breaches. However, the usefulness of this methodology is significantly limited as the number of possible locations may grow to encompass the entire NFZ perimeter as time between sightings increases. In our simulation, detailed in Section 3, growth, \( \lambda \), from time step \( t \) to \( t+1 \), a single time step, is bounded by

\[
\sum_{\forall \{P(L, t) = 0\}} N(L, t) \geq \lambda \geq \sum_{\forall \{P(L, t) = 0\}} N(L, t) + 1 \quad (4)
\]

2.2.2 Utilizing historical encounters. In order for guards to improve their behavior, and thus decrease the number and depth of intruder penetrations into the NFZ, a temporal learning method is incorporated. The type of learning method utilized is reinforcement learning where an agent’s goal is to maximize its reward for a particular state based on historical encounters. Each historical encounter or episode is used to increase knowledge about the probable location of the intruder. These historical observations allow us to extend the idea of possible location, defined by the possibility map, to include likelihood, defined by a likelihood map.

The likelihood that an intruder is located at one location over another is a direct result of the intruder’s strategy model. Since this model is not known a priori, guards must use available observational data as clues to improve their own behaviors. Likelihood, in this work, utilizes sets of breaching locations, including entry and exit, as observations to learn patterns within the underlying strategy model of the intruder. We assume that the intruder exhibits behavior patterns and is not randomly selecting paths that breach the NFZ at indiscriminate times and locations, thus creating the conditions necessary for learning.

Although the method of determining likelihood from historical observations is a well-studied area, the effectiveness is highly dependent on variable selection and implementation. Poor selection of variables generally results in poor or un-useful data that may lead to incorrect assumptions, poor decisions, and wasted computational time. This approach utilizes two unique quantities based solely on breaching entry points and breaching exit points. These variables were selected for two reasons. First, the state space is bound by the perimeter distance of the NFZ. This allows for limited storage requirements and simplifies calculations. Second, these locations are the locations of interest in terms of preventing NFZ penetration. They represent the endpoints of a historical path by the intruder and may reveal information about the underlying strategy of the intruder.

Specifically, for each pair of sequential exit and entry points this algorithm extracts the perimeter distance from the new point of entry and previous exit point, denoted as \( \theta_m \), as well as temporal information in the form of the number of time steps that occurred between breaches, denoted as \( \theta_n \). The set of all observations for an intruder is denoted as \( \Theta \).

Given the last breaching exit point, \( L_B \), and the amount of time that the intruder has been outside of the NFZ, \( \Delta t \), the likelihood of a location, \( L \), containing the intruder, \( L^1_L(t) \), is determined with the following

\[
L^1_L(t) = P(L, t) \ast \left( 1 + \sum_{\theta \in \Theta} Ach(L, \theta_m + L_B, \theta_n - \Delta t) \right) \quad (6)
\]

where

\[
Ach(L_1, L_2, n) = \begin{cases} 
1, & \text{Dist}(L_1, L_2) \leq n \\
0, & \text{otherwise}
\end{cases} \quad (7)
\]

and \( \text{Dist}(L_1, L_2) \) is the distance of the shortest possible path between \( L_1 \) and \( L_2 \).

This approach simulates a type of counter at each possible location of the intruder. This pseudo-counter represents the number of historical observations that could inhabit location \( L \) at time \( t \). Using equation (6) to define each location’s likelihood, a discretized map, or likelihood map, can be generated.

As an example, consider a single historical observation that shows an intruder reentering the NFZ six units to the left of its exit point after 10 time steps (i.e. \( \theta_m = -6 \) and \( \theta_n = 10 \)). Figure 1 details the values \( L^1_L(t) \) at three arbitrary time steps, \( t = \{1, 4, 8\} \), where \( E_X \) and \( E_E \) represent the intruder’s last exit point and hypothetical entry points, \( \theta_m + L_B \), respectively. As can be seen in Figure 1, locations where \( L^1_L(t) = 1 \) are possible locations that have no corresponding historical data. All locations where \( L^1_L(t) > 1 \) have \( L^1_L(t) - 1 \) distinct historical observations that correspond to that location. Figure 1 details a combination of the possibility region, Section 2.2.1, with information extracted from historical encounters with an intruder.
The likelihood map is recalculated at each time step and is used by each guard to determine its own optimal location within the NFZ. This optimal position is greedy defined as the location within the NFZ that has the highest instantaneous reward. An individual location’s reward value is equal to the sum of all values within the likelihood map that would fall within that guard’s sensor range if it was located at that position. Each guard then selects an action that will move it towards its optimal location.

In order to prevent guards from selecting the same optimal location, the operation of selecting an action is performed asynchronously. Once a guard selects its optimal location the likelihood map is updated to remove all rewards that would be captured if that guard was located at its optimal location. This not only prevents guards from crowding at high reward areas while leaving low reward areas unprotected.

2.2.3 Knowledge decay. Although the method describe in Section 2.2.2 may prove significantly useful for intruders with a static strategy, the assumption that an intruder will not modify and adapt its strategy is somewhat unrealistic. In an effort to account for intruders that adapt or drastically change strategies, a memory decay methodology was implemented. By decaying the influence of select historical data, guards are able to account for new intruder strategies.

One of the most common methods of memory decay is time-based decay. This method applies a weighting effect to memory based on age. In this type of implementation older memory is given less weight and is thus less influential on current decisions. Although this method has proved extremely useful in many learning problems it has several drawbacks within the scope of our task.

First, traditional time-based decay methods assume that memory, or in our case historical data, is time invariant. Consequently new data are constantly replenishing decaying information. In this work, new data are only available if the guards are unsuccessful at keeping intruders out of the NFZ. As such, in instances where the guards are successfully preventing penetration a traditional time-based decay method simply reduces the amount of memory and may have significantly negative effects on the guard’s behavior.

A second disadvantage is that traditional time-based decay typically assumes that the oldest memory is the least valuable. If this were the case, the time invariant issue described above could be overcome by using historical data as a time line. Memory would decay as a function of the frequency of newer historical data points. Although potentially useful in many problems, our observational data is directly related to effectiveness of our current strategy. As our current strategy is based on the sequential addition of historical observations, a simple decay of the oldest historical data may diminish or invalidate newer historical data.

Given the above problems, it was decided that the importance of historical observations should be weighted based on the effectiveness of the current strategy. The effectiveness of a particular strategy is proportional to the amount of time between consecutive attacks. Let $\Theta = \{H_1, H_2, \ldots, H_t\}$ where $H$ represents a distinct historical observation and let $W = \{w_1, w_2, \ldots, w_t\}$ be the set of weights for those observations. When a new penetration occurs, denoted as $O$, we can assume that the guards’ strategy was effective against any historical encounter where $H(\theta_n) < O(\theta_n)$. However, the current strategy is not likely to be effective against historical encounters where $H(\theta_n) \geq O(\theta_n)$. Consequently, all $\Theta$ where $H(\theta_n) < O(\theta_n)$ have unchanged weights and all $\Theta$ where $H(\theta_n) \geq O(\theta_n)$ receive discounted weights, i.e. the weight is multiplied by a discount factor in the range $(0,1)$. Assuming $O$ is a new distinct historical observation, it is added to $\Theta$ with $w = 1$, otherwise the equivalent $H$’s weight is set equal to one. In order for $W$ values to influence the likelihood map equation (7) is modified to

\[
Ach(L_1, L_2, n) = \begin{cases} 
  w_d \times \text{Dist}(L_1, L_2) \leq n \\
  0, & \text{otherwise}
\end{cases}
\]
where $w_d$ represents the $d^{th}$ element of $\Theta$.

It should be noted that the above decay implementation never removes historical data from memory. Although it is not addressed in this work, an intruder may switch between several intrusion strategies. If the decay algorithm removes historical data from memory, guards would be forced to relearn previously seen intruder strategies.

### 2.3 Intruder behaviors

In order to perform a comparison study to evaluate the significance of incorporating historical observations of the intruder into the guard’s behavior model, several different intruder behaviors were implemented. The various behaviors allowed us to develop comparison metrics that can be used for inspection, redesign, and error checking. Furthermore, the ability of the guard to adapt to various intruders demonstrates the robustness of the approach.

Each of the behaviors below dictates how the agent will behave when in a particular state and remains the same throughout a simulation run, regardless of conditions. Sections 2.3.1, 2.3.2, and 2.3.3 outline the details of the intruder agent behaviors.

#### 2.3.1 Simple intruder

The “simple” intruder contains the most basic underlying strategy. This intruder follows the shortest distance path to the center of the goal until it is within direct sensing range of a guard agent. Once the intruder is directly sensed by a guard it will follow the longest distance from the center of the NFZ along the $x$- or $y$-axis until it reaches the simulation or world boundary. Once the world boundary is reached, the intruder will again pursue the center of the NFZ. Given the constraints of this strategy, the intruder’s behavior will quickly converge on a single entry and exit point into the NFZ.

#### 2.3.2 Rotating intruder

Much like the simple intruder, the rotating intruder utilizes the shortest distance path to pursue the center of the NFZ. This intruder preemptively looks forward a single step to determine if its next action will take it within sensing range of a guard. This assumes that the sensing range of the intruder, $S_I$, is equal to $S_G + 1$. If it is determined that the next location on the intruder’s path is within sensor range of a guard, the intruder will perform a predefined retreat and circle behavior in an attempt to go around the guard. The strategy has a distinct advantage in that the intruder may perform multiple retreats and circling behaviors without alerting the guard(s) to its position. Consequently, the area of possible locations being calculated by the guard(s) can grow extensively large.

It should be noted that if intruders and guards were given equal sensor ranges, the guards would see the intruder on every intrusion attempt. This would provide the guards with significantly increased feedback and in certain intrusion behaviors even allow for the exact path taken by the intruder to be calculated. As this work focuses on a feedback starved environment, the authors chose to provide intruders with greater sensor range, i.e. $S_G + 1$.

#### 2.3.3 Devious intruder

The “devious” intruder incorporates three distinct strategies; attacking, escaping, and unsticking. The attacking behavior for this intruder selects the action that maximizes distance gain to the center of the NFZ that will not place the intruder within sensing range of a guard. Like the rotation intruder, this intruder has sensor range $S_I = S_G + 1$ and can thus see guards when he is within one unit of their sensor range. This allows the intruder to attempt to slide around the perimeter of a guard’s sensor range undetected. Unlike the rotating intruder, the devious intruder will generally keep the guards within sensor range when attacking.

---

**Algorithm 1.** Simple intruder behavior.

**Input:** The internal state of the intruder (self), initial state = attack

**Output:** The desired movement action of the intruder

```plaintext
if (self.state == attack) //am I in ‘attack’ mode
    desired_action = SHORT_DIST(self, goal) //take move that generates the shortest distance to the goal
    if (SEEN(self.pos) == TRUE) //if a guard can see me
        self.state = run //set mode to ‘run’

if (self.state == run) //am I in ‘run’ mode
    desired_action = LONG_XYDIST(self, goal) //take move on x or y axis that generates the longest distance from the goal
    if (BORDER(self.pos + desired_action, world) == TRUE) //will current action take me to a boundary
        self.state = attack //set mode to ‘attack’
```

---
In the event that the intruder finds that all available actions will result in a location that is covered by a guard’s sensor, the intruder will consider itself trapped by multiple guards and will select an action that maximizes its distance from the center of mass of all guards within the intruder’s sensor range.

**Algorithm 2.** Rotating intruder behavior.

**Input:** The internal state of the intruder (self), initial state = attack  
**Output:** The desired movement action of the intruder

```
desired_action = NULL  //stay still
if (self.state == attack)  //am I in ‘attack’ mode
    desired_action = SHORT_DIST(self, goal)  //take move that generates the shortest distance to the goal

if (self.state == rotate)  //am I in ‘rotate’ mode
    desired_action = ROTATE_RIGHT(self, goal)  //take move that rotates my position to the right
    if (ROTATE_DIST(self) > rotate_setpoint)  //have I moved my predefined right hand rotations
        self.state == attack  //set mode to ‘attack’

if (SEEN(self.pos + desired_action) == TRUE)  //will my current action cause me to be seen by a guard
    self.state = run  //set mode to ‘run’
    RESET(rotate_setpoint)  //re-initialize the predetermined rotation length

if (self.state == run)  //am I in ‘run’ mode
    desired_action = LONG_DIST(self, goal)  //take move that generates the longest distance to the goal
    if (DIST(self.pos + desired_action, NFZ) > nfz_setpoint)  //am I far enough from the NFZ
        self.state = rotate  //set mode to ‘rotate’
```

**Algorithm 3.** Devious intruder behavior.

**Input:** The internal state of the intruder (self)  
**Output:** The desired movement of the intruder

```
desired_actions= SHORT_SAFE_DIST(self, goal)  //take move that is the shortest distance to the goal  
//that does not pass into a guard’s sensor range

if (EMPTY(desired_actions) & self.state == attack)  //if all moves pass into guard’s range I’m surrounded
    self.state = surrounded
else if (SIZE(desired_action > 1)  //if there is more than one “best” action
    desired_action = RANDOM(desired_actions)

if(self.state == surrounded)
    desired_action = FURTHEST_DIST(GUARDS_SEEN(self))  //take move that maximizes the distances  
//from guards
    if DIST_MOV(self) > dist_setpoint  //if I’ve ran far enough try attacking again
        self.state = attack

if(AREA_MOV(self, time) < mov_threshold)  //if I’ve been stuck in the same area for a long period of time
    self.state = stuck
    set_corner = FAR_CORNER(self, world)  //find the farthest corner from my current position

if (self.state == stuck)  //if I’m stuck
    desired_action= SHORT_SAFE_DIST(self, set_corner)  //move = shortest distance to the corner that  
//does not pass into a guard’s range
    if (EMPTY(desired_action))  //if all moves pass through a guard’s range; brute force a path
        SHORT_DIST (self, set_corner)
```

In the event that the intruder finds that all available actions will result in a location that is covered by a guard’s sensor, the intruder will consider itself trapped by multiple guards and will select an action that maximizes its distance from the center of mass of all guards within the intruder’s sensor range.
Last, if this intruder finds that it has had been confined to a small, a priori defined size, region for a predetermined amount of time the intruder will consider itself stuck. In this instance the intruder will pursue the furthest corner of the simulation from its current position while attempting to avoid being sensed by guards.

Unlike intruders described in Sections 2.3.1 and 2.3.2, the devious intruder is not entirely discriminate. In any instance that this intruder finds that multiple optimal actions exist, the intruder will randomly pick one of the valid actions.

3. Simulation environment

In order to test our guarding and intruder algorithms, a custom simulation environment was created in Matlab, Figure 2. This simulation tool discretizes a two-dimensional world into \( m_O \times m_O \) cells, where \( m_O \in \mathbb{Z}^+ \), containing three distinct zones; a goal zone (GZ), a NFZ, and open airspace zone (OAZ). These zones are characterized by:

- A NFZ is an \( m_N \times m_N \) area, where \( m_N < m_O \in \mathbb{Z}^+ \), within the OAZ that designates the range limitations of all guarding agents and signifies cells that can alert guarding agents to the presence of an intruder.
- A GZ is an \( m_T \times m_T \) area, where \( m_T < m_N \in \mathbb{Z}^+ \), within the NFZ that designates the \( P_I \) that guards are willing to tolerate. A penetration into a GZ concludes a single simulation run.
- The OAZ represents all space that is not covered by the NFZ. The OAZ designates the locations where an intruder can operate undetected by guarding agents. The only exception to this is if the OAZ location is directly observable by a guard (i.e. within sensing range, \( S_G \)).

As mentioned above, the detection range of the intruder can be extended outside of the NFZ by the guard’s sensing range, \( S_G \). This simulation constrains the sensor range to a circular area centered on the guard(s) and is considered to be noise free. Beyond range, the guard’s sensor provides significant coverage for areas outside of the NFZ. More specifically, sensed area can be extended by a minimum of \( \pi S_G^2/2 \) per guard up to a maximum of \( 2m_N/S_G \) guarding agents. Beyond \( 2m_N/S_G \) guarding agents, the coverage gain becomes discounted due to perimeter crowding of the guards.

It should be noted that although sensors are portrayed as radial, in implementation all sensors operate as rectilinear. This conversion is done by allowing a location \( L \) to only be sensed if its midpoint location is within the radial range of the sensor.

It should be noted that all guard agents have a single behavior when the intruder is located within the NFZ. The goal of this behavior is to place the guard between the intruder and the GZ. Once the intruder’s path has been blocked by the guard, the guard will then chase the intruder out of the NFZ. This behavior is also activated if the intruder is outside of the NFZ but is directly sensed by a guard. In this case, only the guards within direct sensor range will chase the intruder and then only within the confines of the NFZ.

4. Results

In order to compare the advantages and disadvantages of attempting to integrate past observations, numerous simulations were performed.

4.1 Base case comparisons

In these experiments learning guards were directly compared with non-learning guards. The base case guarding strategy used for comparison in these experiments utilized possibility maps calculated using the method described in Section 2.2.1. These results were compared with results from learning guard(s) that employed the method described in Section 2.2.2. These specific simulation results utilized a single intruder and two guards operating in a 50 \( \times \) 50 world with a 21 \( \times \) 21 NFZ and a 7 \( \times \) 7 GZ. Note that due to the configuration of the simulation, a penetration depth of eight represents a penetration into the GZ and concludes the simulation. These prematurely ended simulations are fully incorporated into the results without normalization over the number simulation steps.
Experiments were run in batches of simulations with a maximum of 1000 moves per simulation. Learners were allowed to carry historical data between simulations, i.e. continuous learning, but started each run with no initial knowledge as to the whereabouts of the intruder.

As expected, simulations for all three intruders showed increased protection of the NFZ when historical data were included. Figure 3 details the average number of time steps, or turns, that the intruder spent at each penetration depth per simulation. Note that penetration depths are non-cumulative and only represent the intruder’s time spent at a particular depth. These particular results provide several interesting occurrences. First, the results imply that the rotation intruder is simpler for both types of guards to address. This is mainly the result of the guard’s inherent behavior combined with intruder parameter selection, specifically the amplitude of the arc for reentry of the intruder. As the possibility map grows around the NFZ, the guard’s inherent behavior is to spread out in an effort to reduce overall closure distance to all possible intruder reentry points. The rotation behavior naturally follows this growth pattern and as such the guard’s inherent behavior proves to be relatively good defense. If the amplitude of the intruder’s reentry arc was chosen to be particularly steep or flat the inherent behavior of the guards may prove to be less of a natural solution and thus create greater need for intruder prediction.

The second interesting occurrence is the relatively poor gain from learning for the simple intruder. This effect is caused by an overly large possibility map. Recall that the simple intruder repeatedly attacks a position but may spend large amounts of time outside of the NFZ. This behavior not only results in an extremely large possibility map but due to the counting method described in Section 2.2.2, the distribution of each past observation throughout this area is extremely widespread.

The last interesting occurrence is the increased number of average penetrations at depth = 1 for the learning method using the devious intruder. This set of results shows that on average the guards allowed the intruder to breach the NFZ a greater number of times but were in better guarding positions to prevent deep penetration. It should also be noted that the devious intruder was able to prematurely end simulations, by reaching the maximum penetration depth (i.e. depth = 8) for 61% of the base case experiments, see Table 1. The learning case significantly improves the protection of the NFZ.

Figure 3. Comparisons of base and learning guards based on various intruders (simulations = 100, steps per simulation \(\leq 1000\)).
decreased this number to 15%. This resulted in an average increase of ~267 extra moves per simulation, roughly 25% of the maximum number of moves allowed per simulation. Overall, even with the increased average number of breaches, the learning guard was able to reduce breach time of the devious intruder from 15.6% of the time to 8.5%.

It should be noted that the learning case using the devious intruder can reduce the number of breaching occurrences for depth = 1 below that of the base case, but it takes a significant number of observations. Figure 4 compares the average results for the base and learning case for a devious intruder using a batch experiment of 1000 simulations and continuous learning. Batch experiments of 1000 simulations and continuous learning were also performed for both the rotating and simple intruder. These results had negligible variations from batch experiments of 100 simulations.

In an effort to discern how fast guarding efficiency increased, i.e. reduced breach time, base cases were compared with learning cases where only the first simulation run was allowed to learn. As such, only historical data from the first simulation were used to construct behavior responses for the guards for the remaining 99 simulations. Table 1 and Figure 5 detail the comparison results of these experiments.

It should be reiterated that the guards have very little feedback about the intruder. For instance, during the single iteration experiment presented in Figure 5, the guards only observed the intruder 18, 13, and 32 times over the 1000 time steps for the simple, rotating, and devious intruders, respectively. These were the only observations utilized for calculating guard movements over the next 99 simulations. Comparing the results (Table 1) of the batch experiments for single iteration learners and continuous learners shows minimal gains in reduced breach time.

The results in Table 1 show that guarding efficiency can be significantly increased with minimal iterations of learning. This was especially true for our devious intruder which incorporated a somewhat complicated strategy and utilized some randomness. This ensures that if an intruder adapts its strategy, the guards may be able to effectively compensate even in a feedback starved environment.

In an effort to discern the effectiveness of incorporating decay into the learning algorithm, several experiments were performed. Initial experiments tested the rate of decay. As expected, any amount of decay increased the average number of penetrations by the intruder. As decay constants approached one, decreasing the speed of decay, average penetration results converged near algorithms using no decay. Conversely, as decay constants approached zero, increasing the speed of decay, average penetration results converged near algorithms using no learning.

Ultimately decay was implemented in an effort to allow guards to quickly adapt to changing intruder strategies. In an effort to quantify the benefits of decay toward these means several experiments were performed. In these experiments the guards were allowed to continuously learn over 100 simulated trials using a devious intruder. The first experiment, or base case, utilized no decay and the second utilized a decay constant of 0.9. This knowledge was then applied to a rotating intruder and averaged over

Table 1. Summary of results applied over 100 simulations.

<table>
<thead>
<tr>
<th>Intruder type</th>
<th>Method</th>
<th>Iterations learning</th>
<th>Simulations ended by GZ breach</th>
<th>Time in breach</th>
<th>Average time steps/simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>Base</td>
<td>N/A</td>
<td>1%</td>
<td>18.2%</td>
<td>990</td>
</tr>
<tr>
<td>Simple</td>
<td>Learning</td>
<td>I</td>
<td>2%</td>
<td>11.2%</td>
<td>970</td>
</tr>
<tr>
<td>Simple</td>
<td>Learning</td>
<td>Continuous</td>
<td>3%</td>
<td>9.2%</td>
<td>970</td>
</tr>
<tr>
<td>Rotating</td>
<td>Base</td>
<td>N/A</td>
<td>1%</td>
<td>7.2%</td>
<td>990</td>
</tr>
<tr>
<td>Rotating</td>
<td>Learning</td>
<td>I</td>
<td>3%</td>
<td>7.1%</td>
<td>971</td>
</tr>
<tr>
<td>Rotating</td>
<td>Learning</td>
<td>Continuous</td>
<td>2%</td>
<td>5.3%</td>
<td>980</td>
</tr>
<tr>
<td>Devious</td>
<td>Base</td>
<td>N/A</td>
<td>61%</td>
<td>15.6%</td>
<td>596</td>
</tr>
<tr>
<td>Devious</td>
<td>Learning</td>
<td>I</td>
<td>16%</td>
<td>8.9%</td>
<td>845</td>
</tr>
<tr>
<td>Devious</td>
<td>Learning</td>
<td>Continuous</td>
<td>15%</td>
<td>8.5%</td>
<td>863</td>
</tr>
</tbody>
</table>
100 simulations. Note that guards continued to learn during the simulations with the rotating intruder, but each simulation began with only the knowledge collected from the 100 devious intruder trials. Figure 6 details the average number of penetrations at each depth for both the base and decay experiments. These results show that the implemented method of decay allows the guards to prevent additional penetrations that would otherwise have occurred.

5. Discussion and conclusions

In this paper, we presented the initial framework for building entity behavior models for intruders trying to enter a secure region. The proposed approach is implemented online utilizing observations of an intruder over time. The results of this work show that meaningful information can be extracted and exploited, via online learning, from a system where feedback is both rare and inconsistently available. Furthermore, the results show that our method of selective memory decay can allow guarding agents to quickly adapt to changes in the intruder’s attack strategy. This work represents a part of the foundation required for creating online learning entities that must develop strategies in real-time in an information starved environment.

Although it was not addressed in this work, there may be significant gains made by weighting the influence of historical data based on the proximity that the intruder got
to the GZ. This would allow varying sized GZs and NFZs to influence the adaptation of the guard’s behavior. Future work will also include the use of intruders with evolving behaviors, increased guard coordination for system wide optimal movements, and guarding units that employ non-greedy algorithms for movement selection.

**Funding**

This work was supported and partially funded by the Autonomous Systems Division of the Vehicles Technology Directorate at the Army Research Laboratory located in Aberdeen Proving Ground, MD.

**References**


**Author biographies**

**Richard Garcia** is a Senior Research Engineer at Southwest Research Institute (SwRI). He has over 10 years of experience designing, implementing, and validating advanced unmanned systems. His areas of expertise focus on software development and system design. His experience includes both ground and air vehicles with a particular focus in heterogeneous coordination/collaboration and intelligent control. Dr Garcia also has significant experience in connected and cooperating systems. This includes third-party evaluation and testing as well as software development of vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) devices.

**MaryAnne Fields** is the team lead for the Intelligent Control Team of the US Army Research Laboratory Vehicle Technology Directorate’s Autonomous Systems Division. Dr Fields received her PhD in mathematics from Clemson University in 1988. Her research interests include modeling and simulation, planning with uncertain information, robotic behavior algorithms, and multi-agent systems. She is currently working on semantic spatial understanding. She is currently a senior editor for the *Journal of Intelligent and Robotics Systems*. She has served as a reviewer for both the IROS and ICRA conferences and has more than 20 publications.

**Robert Kania** is a Research Electrical Engineer leading the Robotic Technologies for the Ground Vehicle Robotics (GVR) Area of the US Army Tank Automotive Research, Development, and Engineering Center (TERDEK). He has been with the GVR since he was hired in July of 2001 serving as both an engineer and a team leader on various teams down the years. Before TARDEC
he worked in the Auto Industry for almost 10 years for various OEMs in both direct-hire and consulting positions. He graduated in 1997 from Ohio Northern University with degrees in Mechanical Engineer, Mathematics, and Electrical Engineering. In the Fall of 2002 he received a Masters Degree in Systems Engineering from Oakland University (OU) and then continued on at OU and was later awarded his PhD in the summer of 2009.

Laura Barnes received a BS in computer science from Texas Tech University in 2003. She received MS and Ph. degrees in computer science and engineering from the University of South Florida, Tampa, in 2007 and 2008, respectively. From 2003 to 2008 she was a research assistant in the Department of Computer Science in the University of South Florida, performing research in the areas of human–robot interaction, unmanned systems, and swarm control. In addition, she held a fellowship from the Army Research Laboratory and performed research in the Vehicles Technology Directorate. She has held faculty positions in the College of Engineering at the University of Texas Arlington and in the College of Medicine at the University of South Florida. She is currently an Assistant Professor in the Systems and Information Engineering Department at the University of Virginia. Her primary research interests include robotics, intelligent systems, and decision support systems.