A Defender-Aware Attacking Guidance Policy for the TAD Differential Game

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Jacob T. English

December 2020

This thesis titled

A Defender-Aware Attacking Guidance Policy for the TAD Differential Game

by

JACOB T. ENGLISH

has been approved for

the Department of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Jay P. Wilhelm

Assistant Professor of Mechanical Engineering

Mei Wei

Dean, Russ College of Engineering and Technology

Abstract

ENGLISH, JACOB T., M.S., December 2020, Computer Science

A Defender-Aware Attacking Guidance Policy for the TAD Differential Game (65 pp.)

Director of Thesis: Jay P. Wilhelm

Deep reinforcement learning was used to train an agent within the framework of a Markov Decision Process (MDP) to pursue a target, while avoiding a defender, for the Target-Attacker-Defender (TAD) differential game of pursuit and evasion. The aim of this work was to explore the games where the previous attacking guidance methods found in literature failed to capture the Target. The reward function of the MDP presented by this work allowed for an attacking agent to learn a policy that expanded the number of cases where the target is captured beyond the former limit of success through the application of the Twin Delayed Deep Deterministic Policy Gradient algorithm (TD3). The strategy developed using artificial intelligence expands the target capture guidance approach to consider the long-term goal, rather than an instantaneous optimal heading. Initial Target positions within a limited set were considered with fixed values for agent velocities and Attacker and Defender initial positions to evaluate the Attacker's learned behavior in comparison with the optimal point capture guidance laws for target capture in the TAD game.

Dedication

Dedicated to Dad, this work would not have been possible without your support.

Acknowledgments

I would like to thank my advisor, Dr. Wilhelm, for his guidance and enthusiasm in the pursuit of my research. His mentorship in my graduate studies has allowed me to grow in both my technical abilities and critical thinking. I would also like to thank Dr. Bunescu, Dr. Liu, and Dr. Casbeer for serving on my committee. Travis, Jeremy, and Theo, your humor, advice, and friendship made working in the lab a great experience. It was a privilege to work in a research setting with students from another discipline.

To my roommates, Andrew, Nathan, and Abbey living with you is what made Athens home for the past six years. I am thankful for having such a consistent group of friends to share the adventures of college and graduate school with. Brie, I have always appreciated being able to share my excitement in my work with you and your tolerance for my late study and research schedule. Mom, Dad, Abby, and Bobby, your support of my education and pride in my accomplishments has been a driving force behind the work that I have done. It was wonderful being home in the last stages of working on this thesis.

Table of Contents

List of Tables

List of Figures

Figure Page

1 Introduction

1.1 Motivation and Problem Statement

Multi-agent pursuit-evasion combat engagement scenarios present difficult problems for the development of guidance methods in aerospace research [3, 4]. The Target-Attacker-Defender (TAD) differential game is a problem of this type that has been widely studied, involving a target agent, an attacking agent, and a defending agent [3–15]. The goal of the Attacker is to capture the Target, while the goal of the Target and Defender is to ensure the Target's survival of the game. Solutions to the TAD problem have been developed for the game where the headings of the agents are unrestricted such that the attacking agent can not make an effort to avoid the Defender to achieve its goal [7, 8, 10, 16, 17]. Using optimal guidance laws for both the defending team and the Attacker, a decision boundary that predicts the outcome of the engagement has been defined given the initial conditions for each agent [7, 8, 16]. Optimal defense strategies defined by numerical solutions for point capture in the TAD differential game have been proven effective against both the optimal attacking methods and attacking guidance using Pure Pursuit or Proportional Navigation [8, 10, 16, 17]. Introducing reinforcement learning to a version of the TAD game where the Attacker is given the additional objective of avoiding the Defender would resulted in a more successful target capture approach which is not limited by the boundary defined by the optimal guidance laws for the unconstrained problem.

Reinforcement learning has proven to be a powerful tool for exploring complex problems through a data-driven approach, where an agent determines the optimal set of actions to take in its environment through trial and error [1]. Recent developments in machine learning, primarily artificial neural networks, have allowed for rapid improvements in artificial intelligence with applications in reinforcement learning, image

classification, object detection, natural language processing, and other complex tasks [18–22]. Work incorporating artificial neural networks and reinforcement learning algorithms, as seen in the Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [2], has enabled deep reinforcement learning techniques to be applied to control of agents for complex, high-dimensional problems. Formulating the TAD engagement as a reinforcement learning problem with a defender-aware attacking agent trained using the TD3 algorithm may be used to provide a policy that exposes new challenges for the defensive strategies.

Guidance laws for variations of TAD have been developed for games involving a Defender that is faster than the Attacker [23], a constrained-maneuverable Defender [24], and a non-zero capture radius [25], but the case where the Attacker actively avoids the Defender in pursuit of the Target has not yet been studied using artificial intelligence. Deep reinforcement learning methods can be applied to explore this new version of the game through training an attacking agent against the optimal point capture guidance laws for the Target and Defender. The contribution of this work is the framing of the TAD game within a Markov Decision Process (MDP) where the Attacker has the added objective of avoiding the Defender in its pursuit of the Target. The reward function presented incorporates these objectives, resulting in an attacking guidance policy that is achieves wins within and outside of the optimal point capture guidance laws' region of capture when a reinforcement learning algorithm is applied. This thesis aims to show that an attacking guidance method, with the ability to avoid the Defender in the TAD game, is not limited by the previously defined outcome prediction barrier.

1.2 Methods Overview

The TAD game was explored through the development of the defender-aware guidance policy in three phases. The first phase involved formulating TAD as a reinforcement learning problem with a reward function that enabled an attacking agent to learn to win the game on a single case where previous guidance methods would fail. Phase II investigated the agent's performance in training on the full set of cases studied, where different hyperparameter values and neural network structures were tested. In the final phase, the agent was trained further using the configuration determined in the previous phase and its performance was compared to the attacking point capture guidance law. The following sections summarize the four phases.

1.3 Phase I Objective: Formulate the Target-Attacker-Defender engagement as a reinforcement learning problem and develop reward function.

The state transition dynamics were established using the optimal point capture guidance law for the defending team and simple motion dynamics in a Markov Decision Process (MDP). Reward functions were examined for a single TAD game to enable the agent in learning to capture the Target in the presence of the optimal point capture defensive strategy.

1.4 Phase II Objective: Tune TD3 hyperparameters and neural network structure.

The attacking agent was trained on the set of the TAD engagements defined by a grid of Target initial locations to tune hyperparemeters and neural netwoks of TD3.

1.5 Phase III Objective: Train the attacking agent further to yield the final policy and compare with attacking point capture guidance law.

The agent was trained against the defensive strategy with the best performing configurations to develop the defender-aware policy. The learned behavior of the agent was compared with the attacking point capture guidance law.

1.6 Summary of Objectives

The deliverable of this thesis was a defender-aware attacking guidance policy, developed using deep reinforcement learning, that outperforms the optimal TAD attacking strategy defined by point capture guidance law. This objective was achieved through the completion of the tasks presented in the phases of this research:

1.6.1 Phase I Tasks

1. Construct a simulation environment with state transition dynamics defined by the differential equations for simple motion and numerical solution for optimal defense in the TAD game

2. Define a reward function and evaluate its success in motivating the attacking agent to intercept the Target in a single engagement where the previous methods would fail

1.6.2 Phase II Tasks

1. Train the agent on a set of games with different initial Target positions

2. Tune the learning algorithm's discount parameter (γ) and exploration noise parameter

3. Experiment with different neural network structures

1.6.3 Phase III Tasks

1. Train the agent to provide the final defender-aware guidance policy
2. Compare the defender-aware guidance policy performance with the attacking point capture strategy

2 Literature Review

2.1 Literature Review Introduction

The TAD game is defined through literature. Game states are divided into two categories depending on a decision surface, where the Target either escapes or is captured. The point capture strategies used to determine optimal headings for the two cases are discussed. Reinforcement learning is introduced with relevant work for the TAD problem. The TD3 algorithm, which improved upon algorithms previously used in solving guidance problems, is described. Continuous control tasks, such as the TAD problem, often require shaped reward signals in guiding a learning agent to achieve its goal. Distance based rewards can be shaped using the Gaussian function to create a gradient that leads the agent to success.

2.2 Target-Attacker-Defender (TAD) Differential Game

Differential games, as defined by Isaacs [26], are restricted by a set of differential equations. In TAD, the Attacker and Defender are agents of the same class and therefore have equal velocities ($V_A = V_D$). The Target velocity (V_T) is slower than that of the Attacker (V_A), otherwise the Defender would not be necessary. By normalizing the velocities using the Attacker velocity, the motion of each agent is defined by the following set of differential equations:

$$\begin{aligned}
\dot{x}_T &= \alpha \cos\phi, \quad & x_T(0) &= x_{T_0} \\
\dot{y}_T &= \alpha \sin\phi, \quad & y_T(0) &= y_{T_0} \\
\dot{x}_A &= \cos\chi, \quad & x_A(0) &= x_{A_0} \\
\dot{y}_A &= \sin\chi, \quad & y_A(0) &= y_{A_0} \\
\dot{x}_D &= \cos\psi, \quad & x_D(0) &= x_{D_0} \\
\dot{y}_D &= \sin\psi, \quad & y_D(0) &= y_{D_0}
\end{aligned} \tag{2.1}$$

where ϕ, χ, and ψ are the headings for the Target, Attacker, and Defender respectively. The variable α is the ratio between the Target velocity and Attacker velocity: $\alpha = \frac{V_T}{V_A} < 1$, which is the scaled velocity of the Target. The model for simple motion in two dimensions defined by Equation 2.1 is standard in studies of the TAD game [3, 8, 12, 15, 25, 27]. While the TAD game is an abstraction of a class of combat engagements likely to occur in a three-dimensional space, analyzing the problem in two dimensions is sufficient as altitude changes in aerial engagements would be relatively small [28]. The optimal point capture guidance laws discussed in this work specifically explored the problem in two-dimensional space.

The TAD engagement was introduced by Boyell [5], where the three-body kinematic relationship between a moving target, an attacking weapon, and a counter-weapon was studied. The closed-form numerical solution for determining the Defender's heading was presented for the case where the counter-weapon was deployed by the Target. Influence of the velocities of the agents on the difficulty of the target defense were discussed. Through analysis of the counter-weapon aimpoint and location of interception, the conditions required for preventing the capture of the Target were defined [6].

Pure pursuit and proportional navigation guidance laws have traditionally been applied as the guidance law for an attacker pursuing a moving target [3, 12, 13, 29, 30]. Pure pursuit guidance involves continuously directing the pursuer's heading to the instantaneous position of its target [29], while methods based on proportional navigation account for the future position of the target through the interceptor-target line of sight [30, 31]. The cooperative strategies developed for a defended target in the research that followed Boyell's work increased the Target's chance of survival when being pursued by an attacker. The traditional target capture guidance laws only consider the states of the attacking agent and the target agent in the determination of the attacking agent's heading. Pure pursuit and proportional navigation are effective methods for target interception, but

the cooperative defensive strategies were designed to counter the attacking behaviors of an adversary that result from using those target capture approaches [8, 15, 16]. This resulted in a need for target pursuit methods that would account for the presence of a defender. Target capture strategies for games involving a defender lead to the definition of regions of success and failure for the Attacker, dependent on the initial conditions of the game and the guidance approaches implemented [8, 13, 16]. The optimal guidance laws for the TAD game with unconstrained headings were presented by Pachter et al. in [15] for both the cooperative defense and the attacking agent, which set the limit for success of the Attacker assuming all agents implement their optimal strategy.

The TAD game has only two outcomes that depend on if an objective can be achieved by one of the players. The objective of the Attacker is to pursue the Target and minimize their separation. The Target and Defender form a team with the objective of preventing the Target's capture through maximizing the distance between the Target and Attacker, while minimizing the distance between the Defender and Attacker at the time of interception. The optimal approaches to precise point capture divide TAD configurations into two regions based on the surface barrier B: the region of escape (R_e) and the region of capture (R_c). The surface B and resulting R_e and R_c are shown in Figure 2.1 for the configuration: $A_0 = (7, 0)$, $D_0 = (-7, 0)$, $\alpha = 0.55$. Within this frame of the game, if the Target were positioned to the right of the surface B in the same region as the Attacker, the Defender would not be able to intercept an attacker playing with the optimal strategy. If the Target were positioned to the left of the surface in the same region as the Defender, it would survive the game when playing optimally.

B is defined as the following set [8]:

$$B = \{(x_A, x_T, y_T) \mid x_A > 0, x_T > 0,\ x_A^2 + \frac{y_T^2}{1-\alpha^2} - \frac{x_T^2}{\alpha^2} = 0\} \tag{2.2}$$

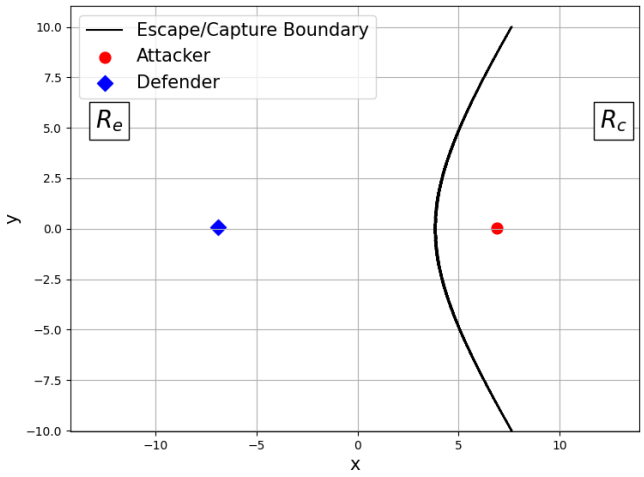


Figure 2.1: Plot of surface B for Attacker position $(7, 0)$, Defender position $(-7, 0)$, $\alpha = 0.55$

The surface B and the optimal strategies use a reduced state space for the game, where the agent coordinates are transformed such that $y_A = 0$, $y_D = 0$, and $x_D = -x_A$. Under optimal play, an interception point is calculated that serves as the aimpoint for the agents, defining a direct path. Deviation from the optimal path by any of the agents will result in suboptimal behaviors and may possibly change the predicted outcome of the engagement. The barrier is dependent on the positions of the Attacker and Defender, and the Target's velocity ratio α.

Games in which the Target initializes in the region of escape are played using the solution to the Active Target Defense Differential Game [15], where the Defender succeeds in intercepting the Attacker and the Target survives. The region of escape includes the set of initial game configurations that belong to:

$$R_e = \{(x_A, x_T, y_T) \mid x_A \geq 0, x_T \geq 0,\ x_A^2 + \frac{y_T^2}{1-\alpha^2} - \frac{x_T^2}{\alpha^2} > 0\} \cup \{(x_A, x_T, y_T) \mid x_A \geq 0, x_T \leq 0\} \tag{2.3}$$

The following quadratic equation (2.4), has two real solutions, y_1 and y_2, where $y_1 \leq y_T \leq y_2$.

$$(1-\alpha^2)y^4 - 2(1-\alpha^2)y_T y^3 + [(1-\alpha^2)y_T^2 + x_A^2 - \alpha^2 x_T^2]y^2 - 2x_A^2 y_T y + x_A^2 y_T^2 = 0 \tag{2.4}$$

Solving for the angles in 2.5 yields the optimal headings for each agent:

$$\begin{aligned}
&\cos\phi^* = \pm\frac{x_T}{\sqrt{x_T^2 + (y_T - y)^2}}, \quad \sin\phi^* = \pm\frac{y_T - y}{\sqrt{x_T^2 + (y_T - y)^2}} \; for \; x_T \neq 0 \\
&\cos\chi^* = -\frac{x_A}{\sqrt{x_A^2 + y^2}}, \qquad \sin\chi^* = \frac{y}{\sqrt{x_A^2 + y^2}} \\
&\cos\psi^* = \frac{x_A}{\sqrt{x_A^2 + y^2}}, \qquad \sin\psi^* = \frac{y}{\sqrt{x_A^2 + y^2}}
\end{aligned} \tag{2.5}$$

If $x_T \leq 0$, $y = y_1$. When $x_T > 0$, $y = y_2$. The optimal heading for the Target, ϕ^*, is solved for using Equation 2.6 if $x_T = 0$.

$$\phi^* = \frac{\sqrt{x_T^2 + (1-\alpha^2)y_T^2}}{\alpha y_T} + \pi/2 \tag{2.6}$$

Optimal play in R_e for $T_0 = (2.30, 4.87)$, $A_0 = (7.0, 0.0)$, $D_0 = (-7.0, 0.0)$, $\alpha = 0.55$ is shown in Figure 2.2.

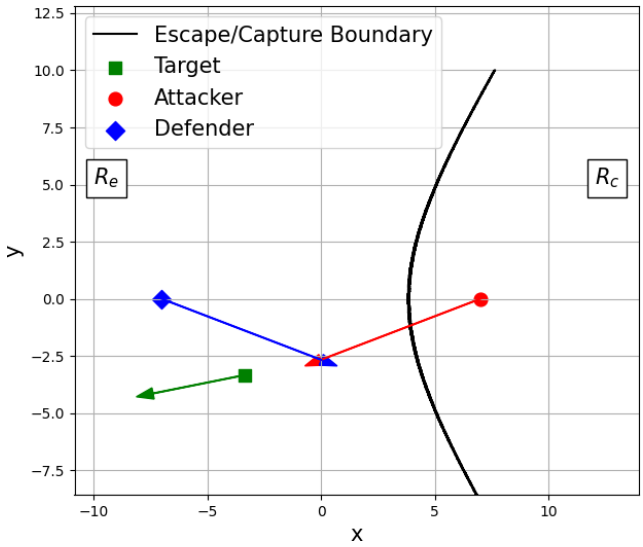


Figure 2.2: Optimal Play in R_e

The TAD capture Game of Degree [8] is played when the Target initial position is in the region of capture, where the Attacker is able to capture the Target. The TAD engagements that initialize within the region of capture belong to the set:

$$R_c = \{(x_A, x_T, y_T) \mid x_A > 0, x_T > 0,\ x_A^2 + \frac{y_T^2}{1-\alpha^2} - \frac{x_T^2}{\alpha^2} < 0\} \tag{2.7}$$

In the TAD capture Game of Degree, the heading of each agent is determined by calculating an aimpoint that is the location where the Target is reach by the Attacker. The aimpoint for a game in R_c is calculated as a point on an Apollonius circle defined by a center point (x_c, y_c) and a radius l. The values for the center and radius are computed using Equations 2.8 and 2.9, respectively.

$$\begin{aligned} x_c &= \frac{1}{1-\alpha^2} x_T - \frac{\alpha^2}{1-\alpha^2} x_A, \\ y_c &= \frac{1}{1-\alpha^2} y_T \end{aligned} \tag{2.8}$$

$$l = \left(\frac{\alpha}{1-\alpha^2}\right)\left(\sqrt{(x_A - x_T)^2 + (y_A - y_T)^2}\right) \tag{2.9}$$

The optimal Attacker-Target interception point on the circle can be calculated as

$$I^* = (l\cos\theta^* + x_c,\ l\sin\theta^* + y_c) \tag{2.10}$$

The parameter θ^* in Equation 2.10 is defined by

$$\theta^* = \arccos Re(v^*) \tag{2.11}$$

where v^* is the solution to Equation 2.12 that minimizes Equation 2.13.

$$\begin{aligned}[x_c y_c + \frac{i}{2}(x_c^2 - x_A^2 - y_c^2)]v^4 + l(y_c + ix_c)v^3 \\ + l(y_c - ix_c)v + x_c y_c - \frac{i}{2}(x_c^2 - x_A^2 - y_c^2) = 0\end{aligned} \tag{2.12}$$

$$\begin{aligned}M(v) = \sqrt{(l\cos v + x_c + x_A)^2 + (l\sin v + y_c)^2} \\ - \sqrt{(l\cos v + x_c - x_A)^2 + (l\sin v + y_c)^2}\end{aligned} \tag{2.13}$$

Using this strategy, the Defender is as close as possible to the Attacker when the Target is captured, allowing the Defender to have the highest potential of intercepting the Attacker when a suboptimal attacking guidance method is being used. Optimal play in R_c for $T_0 = (10.0, 5.55)$, $A_0 = (4.5, 0.0)$, $D_0 = (-4.5, 0.0)$, $\alpha = 0.35$ is shown in Figure 2.3.

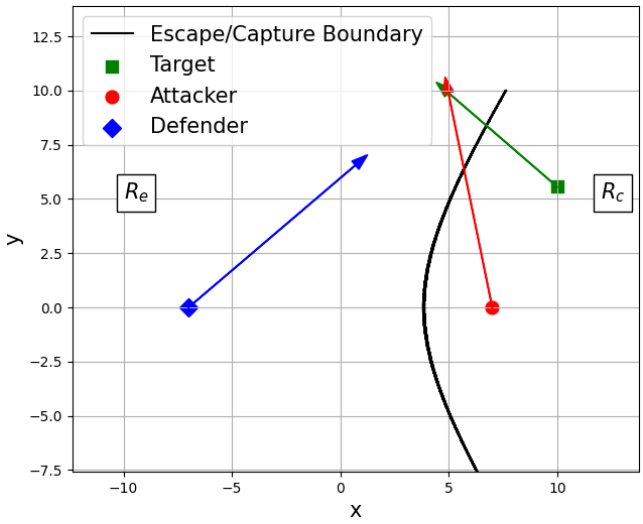


Figure 2.3: Optimal Play in R_c

2.3 Reinforcement Learning

The closed form solutions to the Active Target Defense Differential Game and TAD capture Game of Degree focus on selecting an optimal instantaneous heading, but a method that guides the Attacker by evaluating a game state's potential for future success would be able to take advantage of the Defender's limitations and capture the Target in both regions defined by the optimal unconstrained methods. Reinforcement learning, an area of study within artificial intelligence in which an agent learns to perform a task through repeated interaction with its environment [1], is suitable for developing such a guidance method. As the agent takes actions within the environment, the environment dictates how the state changes and provides a reward signal for the agent to give a measurement of how well it is performing. The reinforcement learning agent develops a policy as it explores the environment, which is a function that maps states to actions, in order to maximize the total reward that it can receive. In the beginning of learning a policy, the agent will take randomized actions to build up its knowledge of the environment.

MDPs are used to represent problems in the reinforcement learning framework [32]. In an MDP, S is the set of possible states the environment can be in and the set A is the action space available to the agent. The state transition dynamics are defined by a function that determines the next state s' that is reached with the reward signal r, given the environment is in state s and the agent takes action a. The optimal policy for an MDP can be derived through dynamic programming techniques to satisfy the Bellman optimality equations [1]. The agent-environment interaction in an MDP is shown in Figure 2.4.

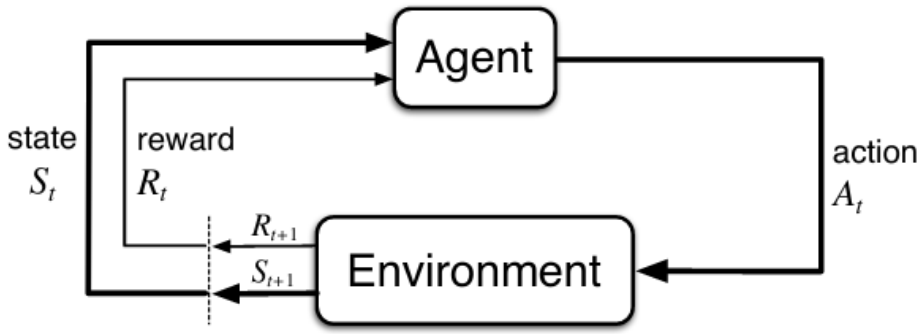


Figure 2.4: Agent-Environment Interaction in a Markov Decision Process [1]

Research has been conducted in evaluating the application of deep reinforcement learning to the development of guidance methods for target pursuit and the TAD differential game. Q-learning was applied to a multi-agent pursuit-evasion game with discrete state and action spaces [33]. A defending agent's guidance policy was learned in [27] for the TAD game in the presence of an attacker using Proportional Navigation. Proximal Policy Optimization (PPO) was used to learn a guidance policy for missile target interception with line of site angle measurements [34]. Policies for guidance of all three agents were learned for the TAD differential game using the Deep Deterministic Policy Gradient algorithm (DDPG) [35]. The cooperative defending policy and attacking policy were trained in alternating batches to match the optimal behaviors of agents following the

guidance laws defined in [3] and [10], respectively. These previous applications of reinforcement learning to games of pursuit and evasion did not involve training an attacker to avoid a defender in capturing its target, and the research where the optimal point capture solutions to the TAD differential game were considered only sought to match their performance.

Q-learning is an off-policy temporal difference control algorithm that uses dynamic programming to directly approximate how good an action is to take in a given state [36]. The Q action-value function is used to act as the policy, where the optimal action is selected based on the maximum Q value for an input state:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a) \tag{2.14}$$

The Q function is updated using Equation 2.15 after the agent takes actions and receives feedback from the environment. In the update equation, η is the update rate that determines how large individual updates are to the action-value estimates. The discount factor γ is used to regulate how much estimates of future states should be factored into the current state's evaluation.

$$Q'(s, a) = Q(s, a) + \eta[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{2.15}$$

The success of deep learning was leveraged to develop high performance reinforcement learning models by Mnih et al. [18], where deep convolutional artificial neural networks were combined with Q-learning to form deep Q-networks (DQNs). DQNs use neural networks to approximate the Q action-value function, which are well suited for complex tasks with high dimensional state and action spaces. Algorithms for continuous control in reinforcement learning combine DQNs with actor-critic methods and deterministic policy gradients to allow actors to learn policies for problems with real-valued state and action parameters. Actor-critic methods decouple the learning of state-values and the policy, allowing for continuous action spaces, in addition to exhibiting

desirable convergence properties [37, 38]. In deterministic policy gradient algorithms, the policy is parameterized by a vector of weights ρ:

$$\pi_\rho(s) = \rho^\top s \tag{2.16}$$

that is updated using the following gradient step:

$$\nabla_\rho J(\rho) = \frac{1}{N} \sum \nabla_a Q(s,a)|_{a=\pi_\rho(s)} \nabla_\rho \pi_\rho(s) \tag{2.17}$$

where ρ is adjusted using stochastic gradient ascent [2, 39, 40]. Parameterizing for updates using stochastic gradient ascent allows the optimal policy function to be approximated as more data about the problem is accumulated through training, to maximize the reward the agent can receive. The Deep Deterministic Policy Gradient algorithm (DDPG) [40] applied the function approximation advantages of neural networks in both its policy parameterization and the action-value function resulting in a robust algorithm that is well suited for problems with continuous action spaces.

The Twin Delayed Deep Deterministic policy gradient algorithm (TD3) [2] is an actor-critic policy gradient algorithm for learning continuous control problems that made improvements to DDPG. TD3, shown in Algorithm 1, employs three methods for improved learning over DDPG: clipped double Q-learning, delayed policy updates, and target policy smoothing. Clipped double Q-learning reduces overestimation bias that is introduced to the Q action-value function [41], an issue that occurs when noisy value estimates accumulate errors through the inclusion of poor estimates over a number of updates [42]. Delaying policy updates to the actor network with the parameter d reduces variance in the learning process by giving more accurate value estimates for TD3 to use in changing the agent's policy. The target policy is smoothed with the addition of random noise, sampled from a distribution $\mathcal{N}$, to prevent overfitting the value estimate. Reducing overestimation bias, delaying policy updates, and policy smoothing all improve the learning process in determining a stable policy function for the given MDP. Exploration of

a given MDP is achieved in TD3 by adding noise (ϵ) to the policy's selected action. Batches of game sequences (s, a, r, s') from sampling the environment are stored in a replay buffer $\mathcal{B}$, that is observed for updating the critic and target neural networks. The authors of TD3 evaluated its performance on a number of tasks from the MuJoCo physics suite in OpenAI Gym (Figure 2.5), where it outperformed DDPG, PPO, and other state of the art continuous control algorithms [2].

Algorithm 1 TD3

Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, and actor network π_ρ with random parameters $\theta_1, \theta_2, \pi_\phi$
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\rho' \leftarrow \rho$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\rho(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$
  Observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\rho'}(s') + \epsilon$, $\epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ρ by the deterministic policy gradient:
    $\nabla_\rho J(\rho) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\rho(s)} \nabla_\rho \pi_\rho(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\rho' \leftarrow \tau\rho + (1 - \tau)\rho'$
  end if
end for

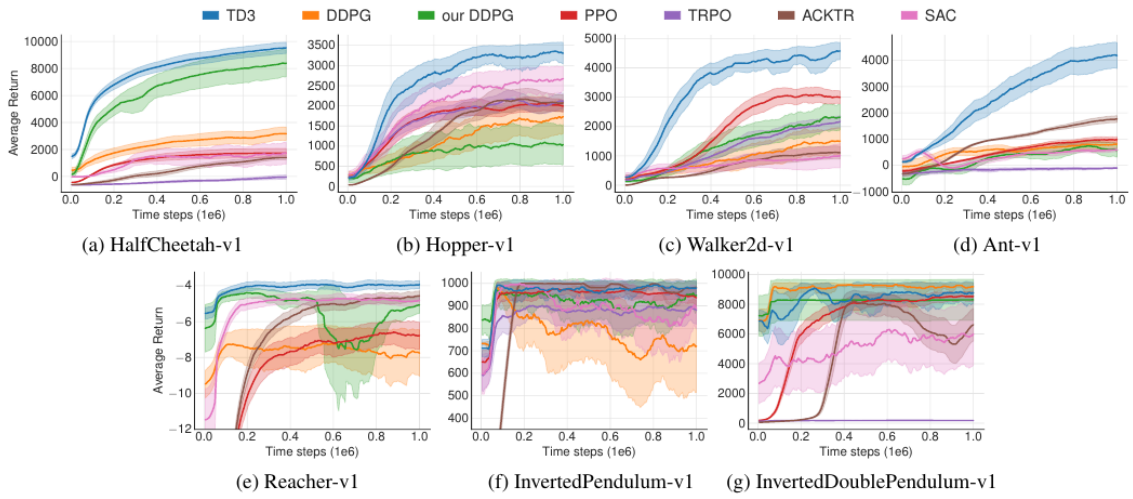


Figure 2.5: Learning Curves for the OpenAI Gym Continuous Control Tasks [2]

In reinforcement learning, the reward function directly impacts the policy that is learned by the agent. Because the agent looks to maximize its total reward from interacting with the environment, it is important to define a reward function that encourages behavior that leads to accomplishing the desired end goal. For complex tasks, binary terminal rewards are often insufficient for providing feedback to the learning agent. Reward shaping addresses this delayed reinforcement problem through providing additional rewards for the agent when it takes actions that approximate the proper behavior [43, 44]. The authors of [45] suggest using a shaping reward inspired by the Gaussian function to act as a progress estimator for goal-directed tasks such as the TAD game:

$$r_{Gaussian}(s) = \beta e^{-\frac{d(s,s_g)^2}{2\sigma^2}} \tag{2.18}$$

The Gaussian function creates an adjustable reward gradient surrounding the goal state s_g, with the scalar β used to tune the intensity of the reward and σ that determines the size of its area of influence. The reward gradient encourages the agent to move to states closer to the goal, as states farther from the goal provide rewards normalized to zero.The Gaussian

reward area is shown in Figure 2.6 for $\beta = 1.0$ and $\sigma = 6.0$ with the goal state located at the origin.

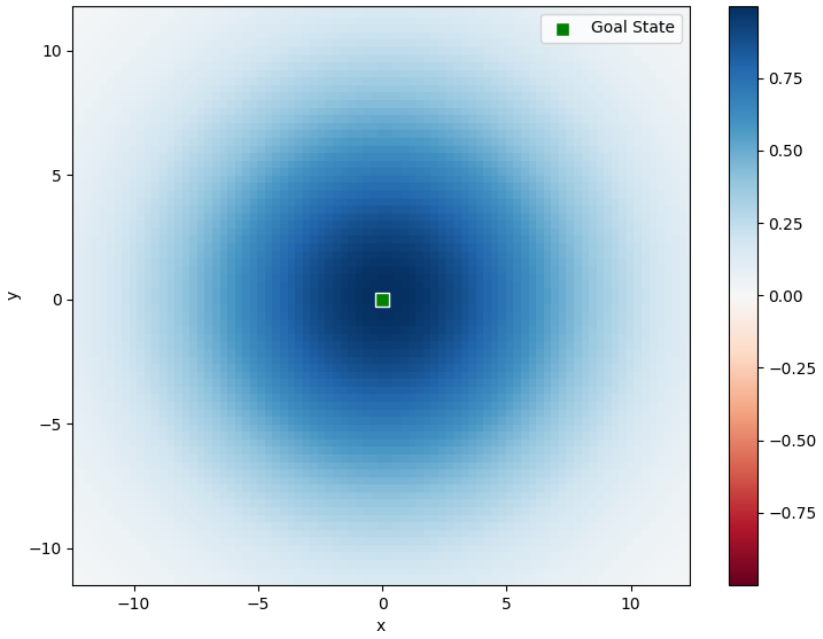


Figure 2.6: Gaussian Reward Area

2.4 Literature Review Summary

Conventional methods for attacking guidance in the TAD game focus on determining instantaneous optimal headings and restrict success to the region of capture determined by the surface barrier B. Reinforcement learning allows agents to learn behaviors that lead to achieving a long-term goal through exploration of a problem space in an MDP. Using TD3 and reward shaping methods, policies can be learned for continuous tasks like the TAD differential game.

3 Methodology

3.1 Introduction to Methodology

Reinforcement learning was used to learn a policy that guides an attacking agent to capture a target in R_c and R_e (where guidance methods from literature have been shown to fail), for the differential game with a turn-rate constrained Target, Attacker, and Defender. The MDP representing the TAD game was constructed using the differential equations for simple motion and point capture guidance laws of the Active Target Defense Differential Game and TAD capture Game of Degree. Reward functions for the game were presented and compared in Phase I. After a reward function was determined that resulted in the desired Attacker behavior, the discount factor and exploration noise were tuned with the neural network structure for TD3 in training batches on the set of Target initializations in Phase II. The learned policy was evaluated in Phase III, where it was compared with the attacking guidance method from literature.

3.2 Phase I: Target-Attacker-Defender Markov Decision Process

The objective of Phase I was to frame the TAD game as an MDP to allow for a reinforcement learning algorithm to be applied in the development of a guidance policy for the Attacker. The state and action spaces were defined to represent the TAD environment. The cooperative defensive strategies and differential equations of motion dictated the transition of one state to the next. Gaussian functions were used in a shaped reward function to learn a successful policy for the Attacker.

3.2.1 State and Action Spaces

The state space for the TAD MDP consists of nine parameters: the coordinates and heading of the Target (x_T, y_T, ϕ), Attacker (x_A, y_A, χ), and Defender (x_D, y_D, ψ), where the

full state of the environment is represented by the vector s:

$$s = (x_T, y_T, \phi, x_A, y_A, \chi, x_D, y_D, \psi) \tag{3.1}$$

which belongs to the state space S:

$$S = \{s \mid s \in \mathbb{R}^9, 0 \leq \phi, \chi, \psi < 2\pi\} \tag{3.2}$$

The action space of the learning agent is a single real-valued parameter that defines the change of heading in radians:

$$a = \Delta\chi \tag{3.3}$$

3.2.2 State Transition Model

Each agent in the TAD MDP transition to their next state with the discrete time interval Δt and equations of simple motion (2.1) such that their next positions are calculated as:

$$\begin{aligned} x'_T &= x_T + \alpha\Delta t\cos\phi, \quad y'_T = y_T + \alpha\Delta t\sin\phi \\ x'_A &= x_A + \Delta t\cos\chi, \qquad y'_A = y_A + \Delta t\sin\chi \\ x'_D &= x_D + \Delta t\cos\psi, \qquad y'_D = y_D + \Delta t\sin\psi \end{aligned} \tag{3.4}$$

The maximum heading change of the agents is restricted by a minimum turn radius, $d_{MinTurn}$, as in [24]. Using the minimum turn radius, the maximum change in headings for the Attacker and Defender are bounded by

$$\Delta\chi_{Max} = \Delta\psi_{Max} = \frac{1.0}{d_{MinTurn}} \tag{3.5}$$

The maximum change in heading for the Target is bounded by the Attacker-Target velocity ratio and the minimum turn radius as follows

$$\Delta\phi_{Max} = \frac{\alpha}{d_{MinTurn}} \tag{3.6}$$

The headings of the Target and Defender, ϕ and ψ respectively, are determined by the optimal point capture solutions from [8] with Equations 2.4 - 2.6 being used when the

Target is in R_e and Equations 2.8 - 2.13 when the Target is in R_c. The Attacker's heading, χ, is adjusted by the action selected by the agent following the policy. In the TAD MDP, a state is considered terminal where either the distance between the Attacker and Target or the Attacker and Defender is less than the capture radius, $d_{Capture}$. More formally, the terminal states belong to the following set, Γ:

$$\Gamma = \{s \mid (\sqrt{(x_A - x_T)^2 + (y_A - y_T)^2} < d_{Capture}) \; or \; (\sqrt{(x_A - x_D)^2 + (y_A - y_D)^2} < d_{Capture})\} \tag{3.7}$$

The terminal set Γ is the union of the sets of states in which the Attacker wins (Γ_{Win}) and loses (Γ_{Loss}):

$$\begin{aligned} \Gamma_{Win} &= \{s \mid \sqrt{(x_A - x_T)^2 + (y_A - y_T)^2} < d_{Capture}\} \\ \Gamma_{Loss} &= \{s \mid \sqrt{(x_A - x_D)^2 + (y_A - y_D)^2} < d_{Capture}\} - \Gamma_{Win} \\ \Gamma &= \Gamma_{Win} \bigcup \Gamma_{Loss} \end{aligned} \tag{3.8}$$

3.2.3 Reward Function

Using a terminal win signal and the reward shaping ideas from [45], the reward function for the TAD MDP is

$$r(s) = \begin{cases} 10 & \text{if } s \in \Gamma_{Win} \\ r_{shaping}(s) & \text{otherwise} \end{cases} \tag{3.9}$$

In order to encourage the attacking agent to learn to pursue the Target and avoid the Defender, Gaussian functions were used to reward decreasing the distance between the Attacker and Target, and to penalize the Attacker for being too close to the Defender. Differences between the Attacker's heading and the angle between the Attacker and Target positions (δ) were penalized to smooth the Attacker's path trajectory. The Attacker was prevented from learning to accumulate rewards far from the Target by subtracting a fixed value (ι) from the shaping reward function. Combining the reward and penalties gives the

full shaping reward:

$$r_{shaping}(s) = \beta_{AT} e^{-\frac{(\sqrt{(x_A - x_T)^2 + (y_A - y_T)^2})^2}{2\sigma_{AT}^2}} - (\beta_{AD} e^{-\frac{(\sqrt{(x_A - x_D)^2 + (y_A - y_D)^2})^2}{2\sigma_{AD}^2}} + \kappa\delta + \iota) \tag{3.10}$$

which simplifies to

$$r_{shaping}(s) = \beta_{AT} e^{-\frac{(x_A - x_T)^2 + (y_A - y_T)^2}{2\sigma_{AT}^2}} - (\beta_{AD} e^{-\frac{(x_A - x_D)^2 + (y_A - y_D)^2}{2\sigma_{AD}^2}} + \kappa\delta + \iota) \tag{3.11}$$

The variables β_{AT}, σ_{AT}, β_{AD}, σ_{AD}, and κ are scalar hyperparameters used to ensure that maximizing the return from the reward function leads to a winning guidance strategy. The shaping reward surface created by Equation 3.11 for an example configuration is shown in Figure 3.1. The surface shows the reward signal the Attacker would receive in a given state on the x-y plane when the Target is located at $(2.30, 4.87)$ the Attacker is located at $(-7.0, 0.0)$. The Attacker's heading offset from the Target (δ) is assumed to be zero for this example with the following hyperparameter values: $\beta_{AT} = 1.0$, $\sigma_{AT} = 10$, $\beta_{AD} = 0.9$, $\sigma_{AD} = 2.0$, $\iota = 0.3$. The shaping reward defined in Equations 3.9 and 3.11 was developed for the game $T_{Init} = (-10.0, 10.0)$ through comparisons to the policies determined by using a terminal win/loss reward and a shaped Attacker-Target distance reward.

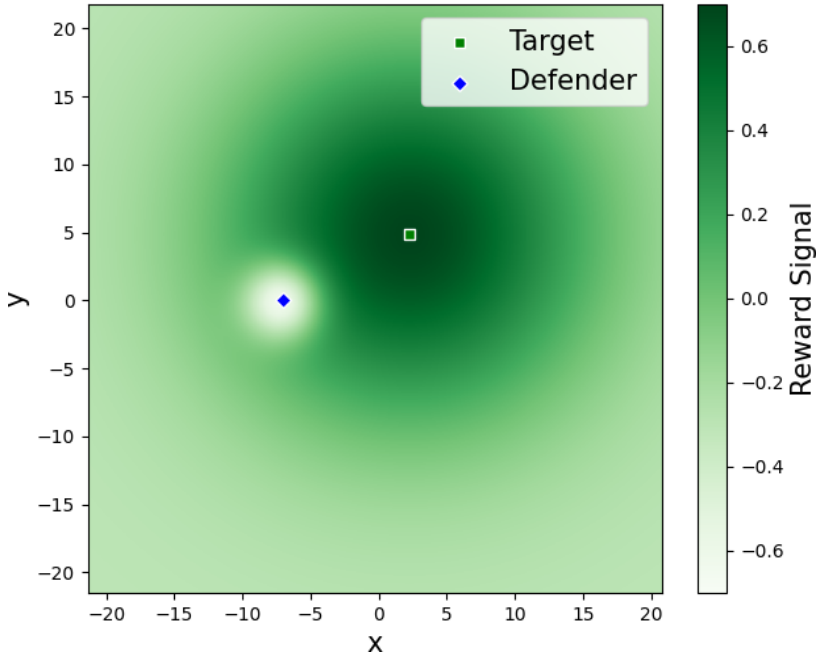


Figure 3.1: Example Shaping Reward Surface

The terminal win/loss reward provides positive feedback for ending the game in a winning state, negative feedback for ending the game in a losing state, and no signal otherwise:

$$r(s) = \begin{cases} 1 & \text{if } s \in \Gamma_{Win} \\ -1 & \text{if } s \in \Gamma_{Loss} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

Without positive feedback from a successful game, the Attacker only received the penalty for its losses which resulted in a policy that caused the Attacker to prolong the game in avoiding the Defender. The learned retreating behavior is shown in Figure 3.2, where the game reached the terminal number of time steps.

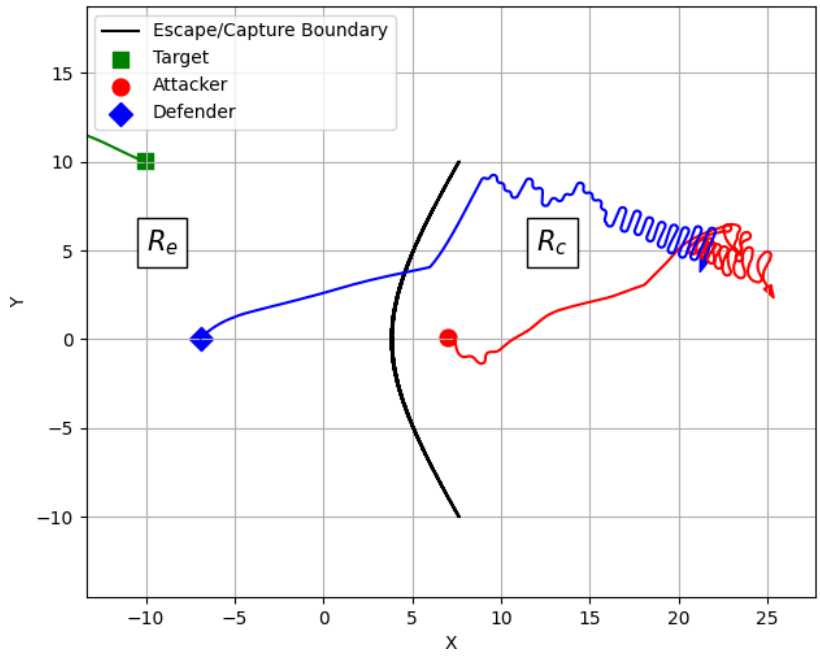


Figure 3.2: Terminal Win/Loss Reward Learned Behavior

The shaped gaussian reward function using only the Attacker-Target separation:

$$r(s) = \beta_{AT} e^{-\frac{(x_A - x_T)^2 + (y_A - y_T)^2}{2\sigma_{AT}^2}} \tag{3.13}$$

provided better results, but still did not lead to a winning policy. Using the learned policy, the attacking agent was able to avoid the Defender inititally, but could not reach the Target. The game where this policy was used is shown in Figure 3.3.

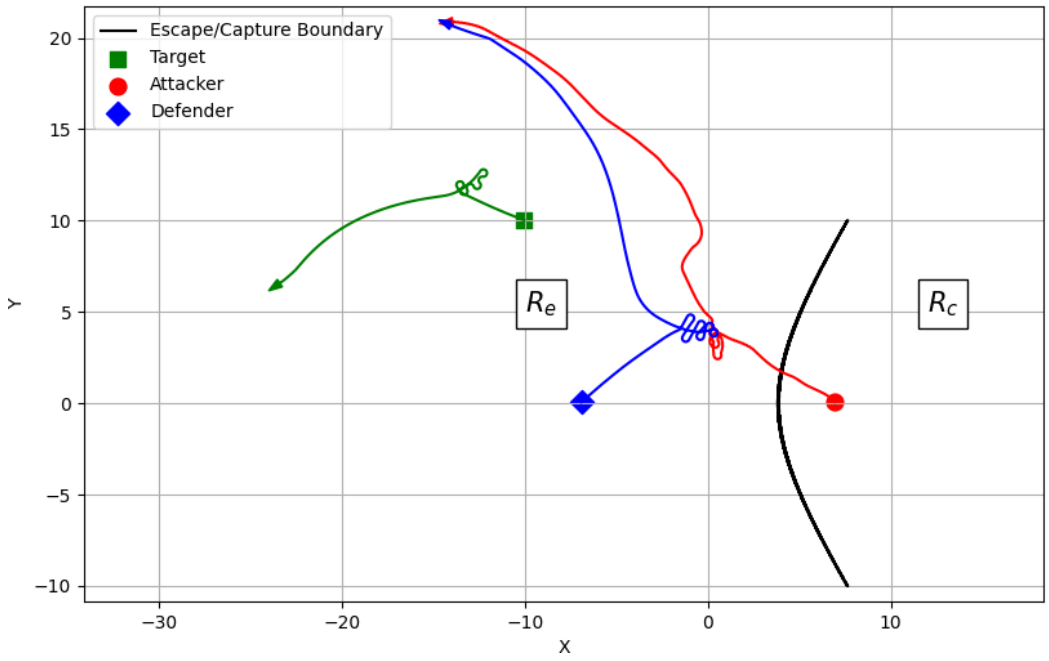


Figure 3.3: Attacker-Target Shaping Reward Learned Behavior

3.2.4 Implementation Details

TD3 was used to train an attacking agent in the TAD MDP environment for the configurations shown in Table 3.1, which were chosen to be consistent with engagement examples found in literature. Given an infinite amount of sampling of the environment in training, reinforcement learning techniques will converge to an optimal policy that satisfies the Bellman optimality equations. In practice, guaranteeing such sampling of the state space is a complex issue [46]. The Attacker was trained on 100 representative games within the problem space defined, where the initial position of the target was varied using a 10×10 grid of equally spaced points such that $x_{T0} \in [-10, 10]$, $y_{T0} \in [-10, 10]$ (Figure 3.4). The initial position of the Attacker was $(7.0, 0.0)$ and the initial position of the Defender was $(-7.0, 0.0)$ for each of the engagements. Training was performed with fixed relations of the Attacker to Defender with the Target initial condition changing to isolate

the problem and allow for fixed relation scaling. Setting the Attacker and Defender at a fixed unit distance with the target being variable allows for any scenario to be applied. The velocity ratio, capture radius, minimum turn radius, and time interval remained unchanged at 0.55, 0.2, 2.0, and 0.1 respectively in this study. These constant parameters were chosen arbitrarily to define a limited problem space for this research; changing any of the values would lead to a different positioning of the escape/capture surface B, and different agent behavior from the learned policy. The starting headings of each agent in the games played were determined by the solutions to the Active Target Defense Differential Game or the TAD capture Game of Degree as appropriate, ensuring that the optimal defensive strategy was initially attainable by the Target and Defender.

Table 3.1: Training Engagement Configurations

Variable	Description	Value
T_0	Target Initial Position	$x_T, y_T \in [-10, 10]$
A_0	Attacker Initial Position	$(7.0, 0.0)$
D_0	Defender Initial Position	$(-7.0, 0.0)$
α	Target/Attacker Velocity Ratio	0.55
$d_{Capture}$	Capture Radius	0.2
$d_{MinTurn}$	Minimum Turn Radius	2.0
Δt	Time Interval	0.1

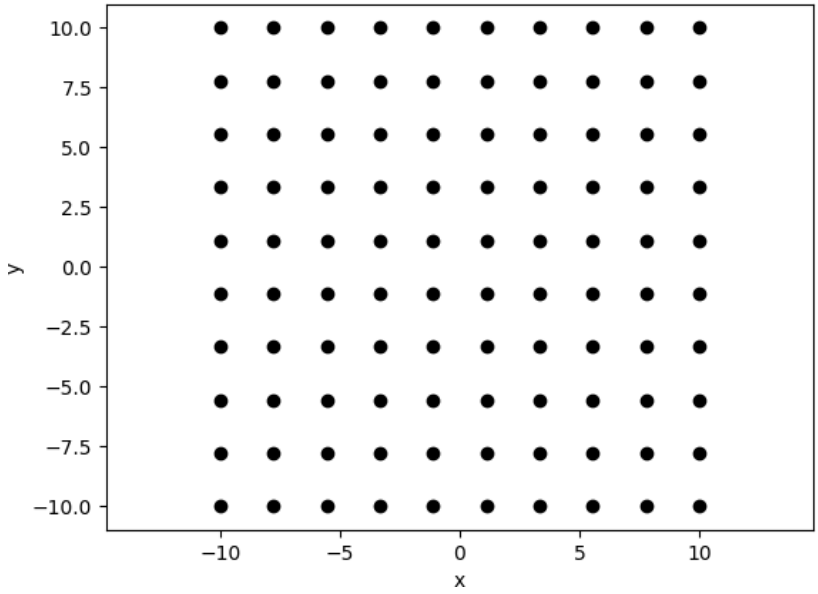


Figure 3.4: Training Dataset of T_0 Cases

At each time step, the control output of the defensive team and the Attacker were calculated and the next state of the game was directed by Equation 3.4 until a terminal state was reached. The input to the TD3 algorithm during training is the full state vector with the resulting reward from the action selected by the policy. When testing the learned policy, the state vector is the only input needed to determine the Attacker's heading change control output.

The simulation environment for the TAD MDP was implemented in Python, and the optimal point capture solutions were computed with the NumPy roots solver. PyTorch was used to create the actor and critic neural networks of the TD3 algorithm. Each hidden layer within the neural networks used the ReLU activation function:

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{otherwise} \end{cases} \tag{3.14}$$

which limits the output of the neurons to a positive range when a positive input signal is received [47]. The activation function for the output layer of the actor network was the

hyperbolic tangent function:

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.15}$$

The hyperbolic tangent function has an output range of $[-1, 1]$, which allowed for the output value to be used as the agent's action by multiplying it with the Attacker's maximum turn rate value.

3.3 Phase II: TD3 Tuning and Experimentation

The objective of Phase II was to determine hyperparameter values and and artificial neural network structure for TD3 that would lead to a successful policy on the games studied. The TD3 algorithm has a number of hyperparameters that can be tuned that change assist the agent in the learning process. In this phase, two hyperparameters were examined: the discount factor (γ) and the exploration noise (ϵ). The discount factor is a measurement of how much the estimated return of future states should be included in the update for the current state-action pair in the value function. In training with TD3, an agent selects random actions to develop the Q action-value function that is used as the policy. The random actions are chosen by adding noise sampled from a normal distribution. It is important that sufficient exploration occurs in training for an optimal policy to be derived. The hyperparameters of both the reward function and the learning algorithm were manually tuned to achieve the desired pursuit and avoidance behaviors of the Attacker by observing the learned policy for cases at the limit of the problem space and overall coverage of the training set.

The Q action-value function from which the policy is derived is developed and represented by artificial neural networks in the TD3 algorithm. The input and output layers of each neural network is determined by the size of the input and action spaces. The layers of nodes within the networks, also known as the hidden layers, abstract the input for nonlinear function approximation. The hidden layer configuration initially used was from

the original implementation of the algorithm and deeper/wider networks were experimented with in this phase.

3.4 Phase III: Extended Training and Evaluation of Learned Policy

The objective of Phase III was to further train the attacking agent using the configuration determined by Phase II and provide an evaluation of the learned policy. The hyperparameter values and network structure for TD3 determined in the previous phase were used to train the Attacker on the same training dataset of Target initial positions. The agent was trained for an extended period of time to result in the maximum win coverage of the problem space. The learned policy's generalization was evaluated on a 100×100 grid of Target initializations within the problem space (Figure 3.5). Resulting Attacker behaviors were observed and compared with the attacking point capture guidance laws.

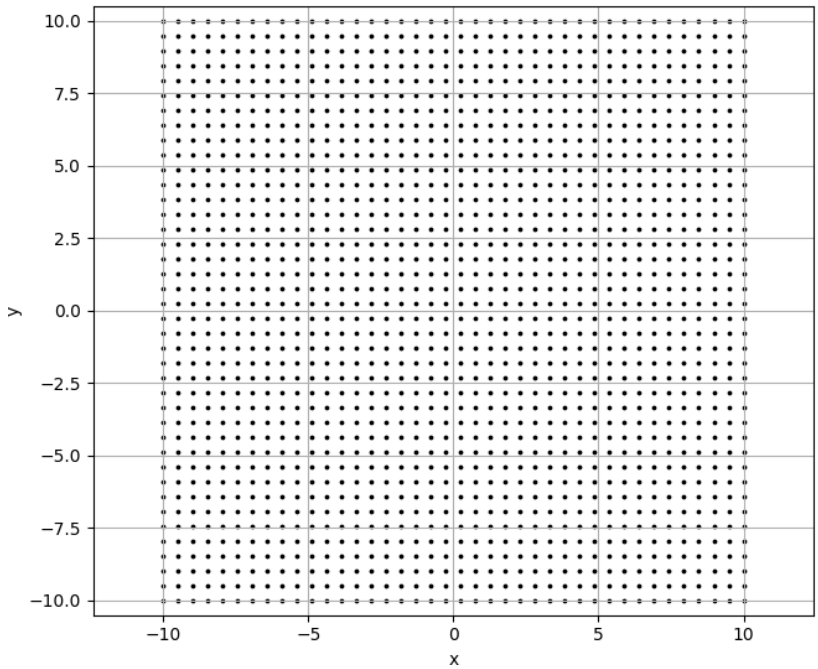


Figure 3.5: Testing Dataset of T_0 Cases

3.5 Summary of Methodology

The TAD model and reward function were developed to allow for a reinforcement learning agent to derive a policy for target capture using the TD3 algorithm and relevant equations were presented. The discount factor and exploration noise hyperparameters of the learning algorithm were selected for experimentation to aid in the learning process. The hidden layer configurations for the artificial neural networks within TD3 were also chosen for experimentation. After determining the final attacking guidance policy, it was compared with the point capture guidance strategies to evaluate the Attacker's performance on a set of games larger than the selected training cases.

4 Results

4.1 Introduction to Results

The MDP was developed and implemented with a distance and heading based shaping reward function in Phase I. The reward function was validated through training the attacking agent on a single case within R_e at the bounds of the studied cases. In Phase II, a successful configuration for TD3 was determined to train the Attacker on the full problem space. The Attacker was further trained to produce the final policy using the configuration from the previous phase in Phase III. The learned policy was compared with the point capture attacking guidance laws and an evaluation of the policy's success was presented.

4.2 Phase I

In Phase I, the TAD MDP was developed. The state transition model and reward function are demonstrated to enable the attacking agent to reach the target for a game initialized within R_e. Training was completed on a CUDA-enabled computer with a GeForce GTX 1050 Ti GPU, an Intel Core i7 CPU, and 32 GB of RAM, which took a total of 40 minutes for a single engagement with $T_{Init} = (-10.0, 10.0)$. The reward returns and win record in training is shown in Figure 4.1.

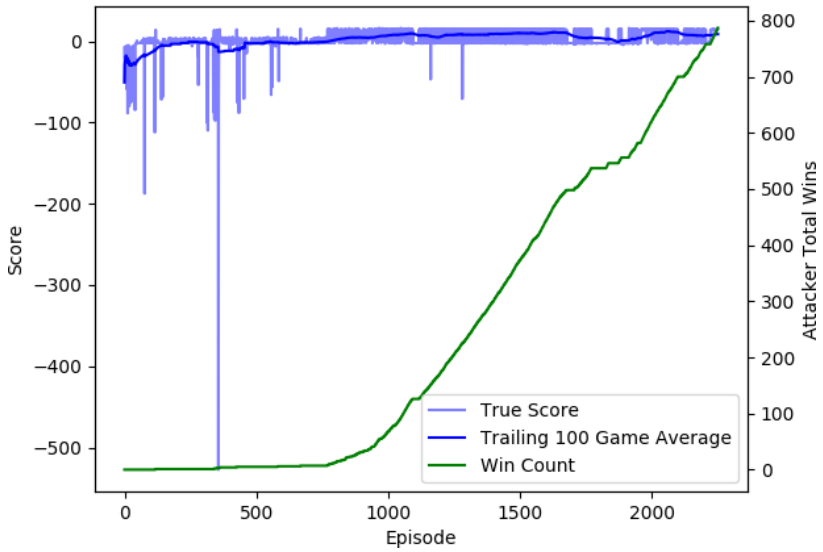


Figure 4.1: Training Scores and Win Record for $T_{Init} = (-10.0, 10.0)$

The Attacker developed a strategy for the TAD game in which it was able to take advantage of the turn rate constraints of the Defender. As the Defender pursues the Attacker, the Attacker baits it into committing to a trajectory. The Attacker then changes course to cause the Defender's optimal interception heading to change in a way that is unachievable due to the restrictions in the version of the game studied. The engagement with $T_{Init} = (-10.0, 10.0)$ involved two instances of the Defender being baited for the Attacker to capture the Target because of the initial distance to the Target and the indirect Attacker path. Each time the Defender is avoided by the Attacker, the game transitions from R_e into R_c causing the optimal defensive strategy to change. This learned behavior of the Attacker from the single case training is shown in Figure 4.2.

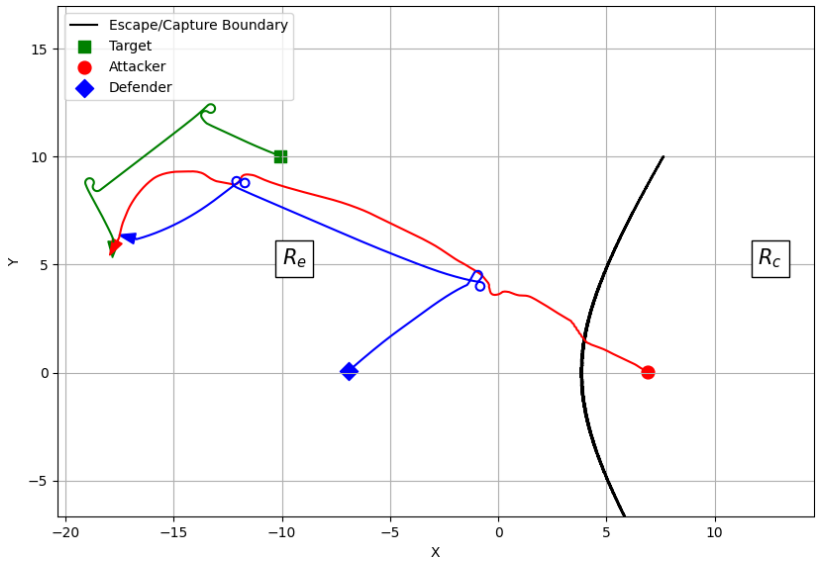


Figure 4.2: Learned Behavior for $T_{Init} = (-10.0, 10.0)$

4.3 Phase II

The second phase of this thesis involved experiments with hyperparameters and artificial neural network structures for the reinforcement learning algorithm, TD3, in training the Attacking agent on a set of 100 training cases in the problem space. The training scores of each configuration were compared after training for $3 * 10^3$ episodes with the results smoothed by a trailing 300 game average.

The discount factor, γ, was tested with values between 0.9000 and 0.9900. The setting $\gamma = 0.9900$ was found to result in the highest score in training (Figure 4.3).

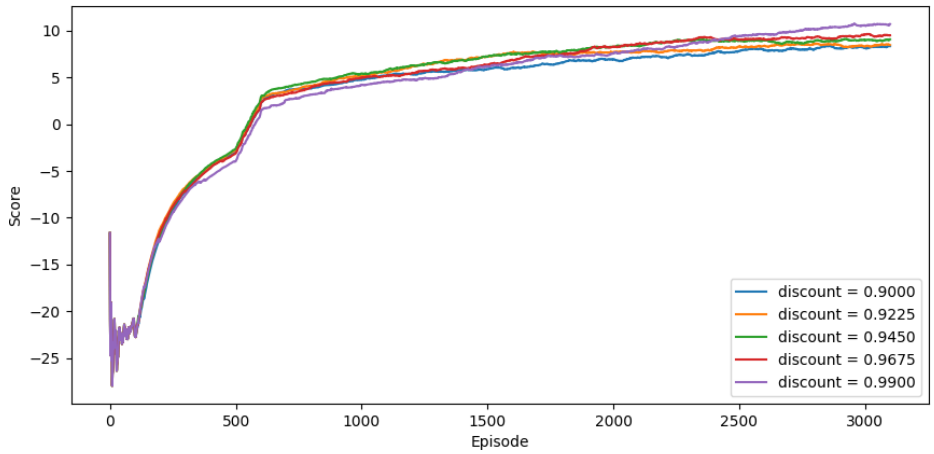


Figure 4.3: Discount Factor Experiments, $\gamma \in [0.9000, 0.9900]$

Because the best performing discount factor was at the limit of the values tested, another bach of values were compared for $\gamma \in [0.9900, 0.9999]$. The discount factor determined from the preceding comparison was consistent in outperforming the other values. The second batch of discount factor training is shown in Figure 4.4.

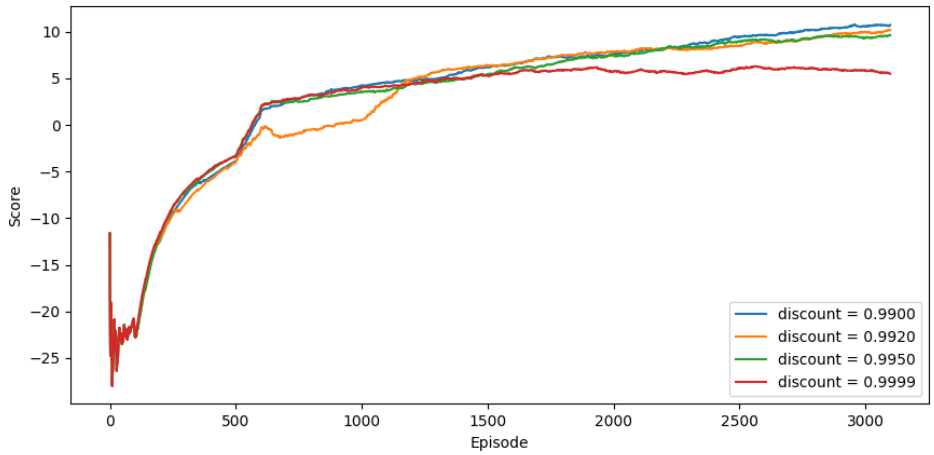


Figure 4.4: Discount Factor Experiments, $\gamma \in [0.9900, 0.9999]$

Values between 0.10 and 0.30 were tested for the exploration noise parameter. It was found that an exploration noise of 0.20 produced the highest scoring policy over the

training period. The training scores resulting from the exploration noise values tested are displayed in Figure 4.5.

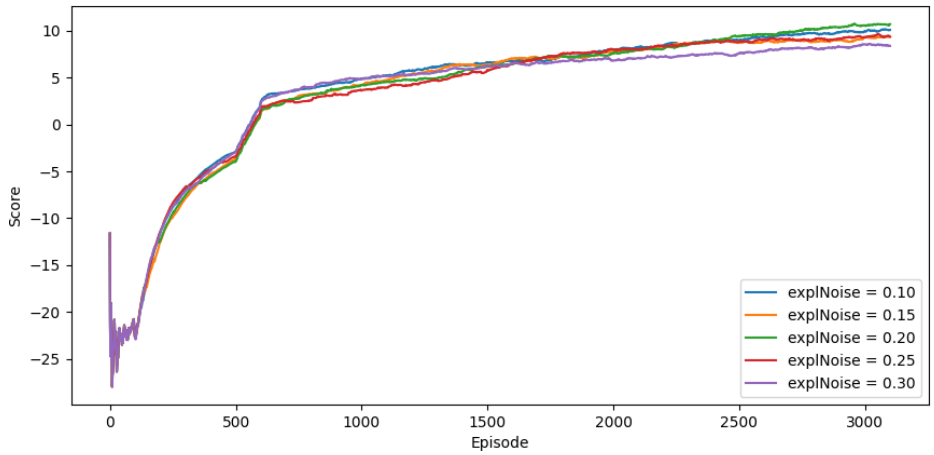


Figure 4.5: Exploration Noise Experiments, $\epsilon \in [0.10, 0.30]$

The original network architectures used in the TD3 implementation had two 256-node hidden layers. The first round of experiments for the network structure tested the depth of the hidden layers, where depths of 2, 3, 4, and 5 layers were examined. The results of training with these configurations is shown in Figure 4.6, where the networks with 4-deep, 256-node hidden layers gained the most return in training.

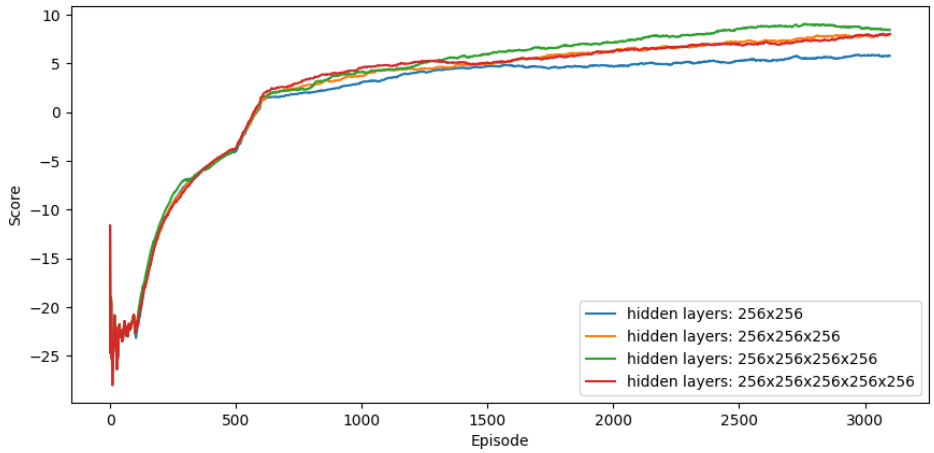


Figure 4.6: Neural Network Structure Experiments, Hidden Layer Depth

Following the hidden layer depth experiment, hidden layers with more nodes were tested. While the slopes of the return in training began to decline for the configurations $256 \times 256 \times 256 \times 256$ and $256 \times 1024 \times 1024 \times 256$, the hidden layer structure $256 \times 512 \times 512 \times 256$ continued to collect higher rewards, as shown in Figure 4.7.

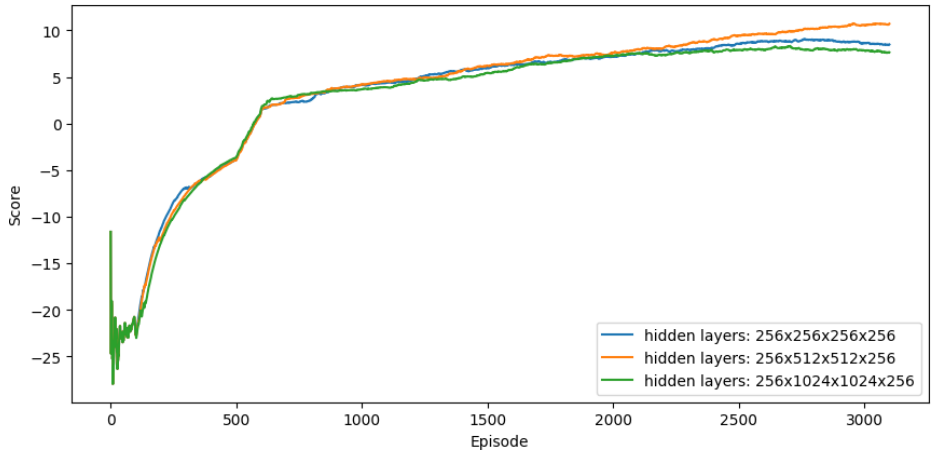


Figure 4.7: Neural Network Structure Experiments, Hidden Layer Width

The experiments in this phase resulted in the configurations for TD3 that were to be used in training the Attacker to learn the final policy.

4.4 Phase III

Training in Phase III was completed in a total of five hours. After training for 1.5e+4 episodes on the 100 initializations for the Target's position, the Attacker learned to beat the defending team in 97% of the cases trained on. The learning agent increased its win rate up to episode 2.5e+3 where the rate became consistent. The trailing 100-game average appears to converge around 1e+4 episodes. The Attacker's score in each game is shown in Figure 4.8 with the running total of wins throughout the training process. Training the Attacker for a larger number of episodes reduced its overall coverage of the Target initial positions. The resulting win/loss coverage for the Attacker on the training cases is shown in Figure 4.9.

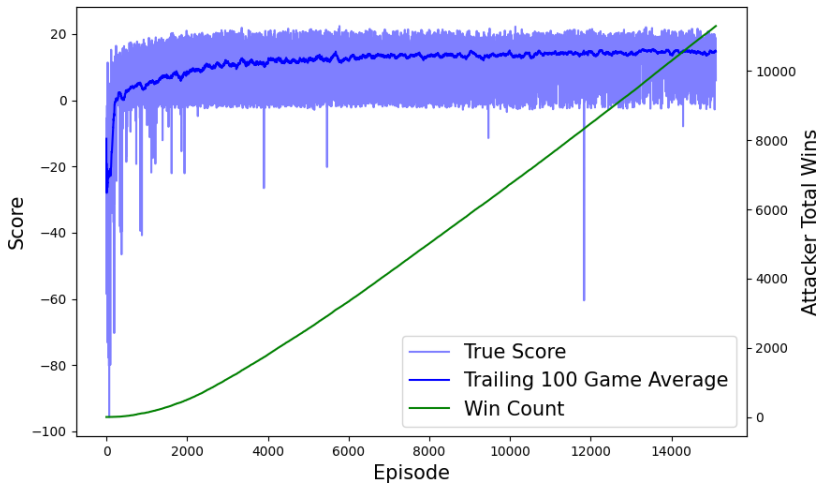


Figure 4.8: Training Scores and Wins

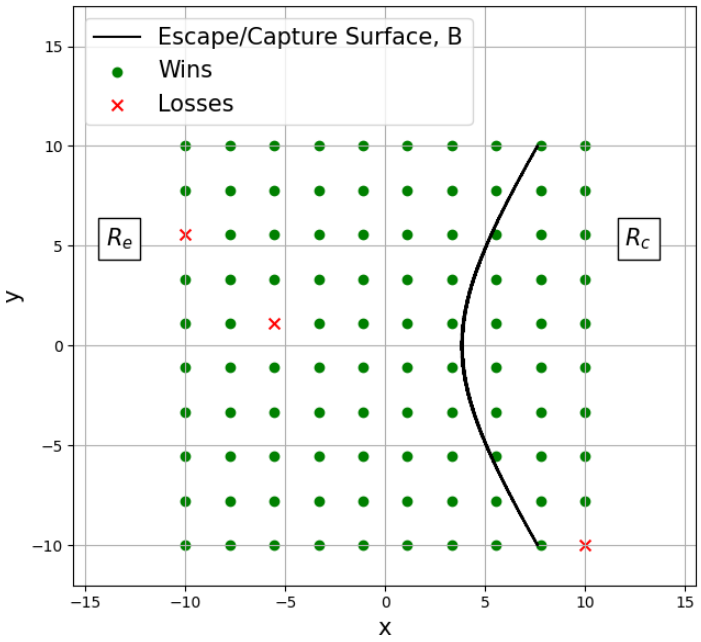


Figure 4.9: Attacker Performance on T_0 Training Cases

The Attacker had similar success when tested on a larger set of games with 1e+3 different Target initial positions to evaluate the guidance policy's generalized performance. The higher resolution of test cases is sufficient for evaluating the learned policy as the initial Target positions are uniformly spaced by the capture radius, $d_{Capture}$. The Attacker intercepted the Target in 88.54% of the tested configurations in Figure 4.10.

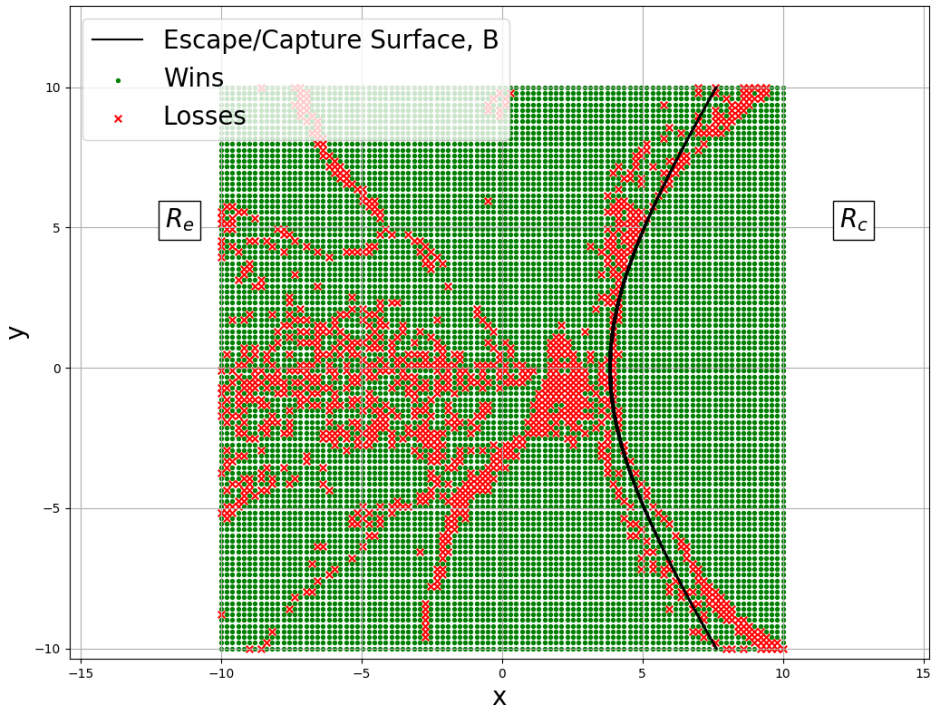
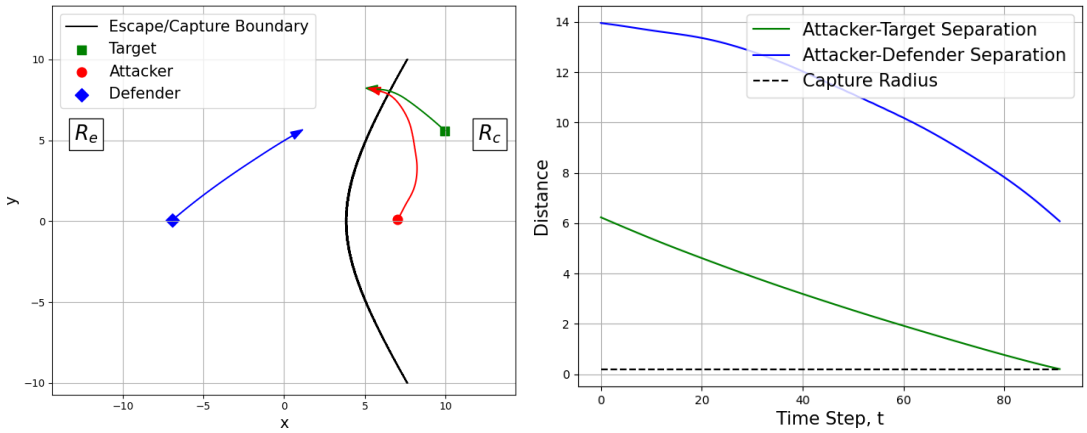


Figure 4.10: Generalized Attacker Performance

The Attacker's learned guidance policy was sufficient for leading to the interception of the Target in engagements within the region of capture, consistent with the performance of the point capture attacking guidance law. The resulting path of the Attacker under the defender-aware guidance policy (a) is presented with the separation of the agents over time (b) for one of these cases in Figure 4.11.

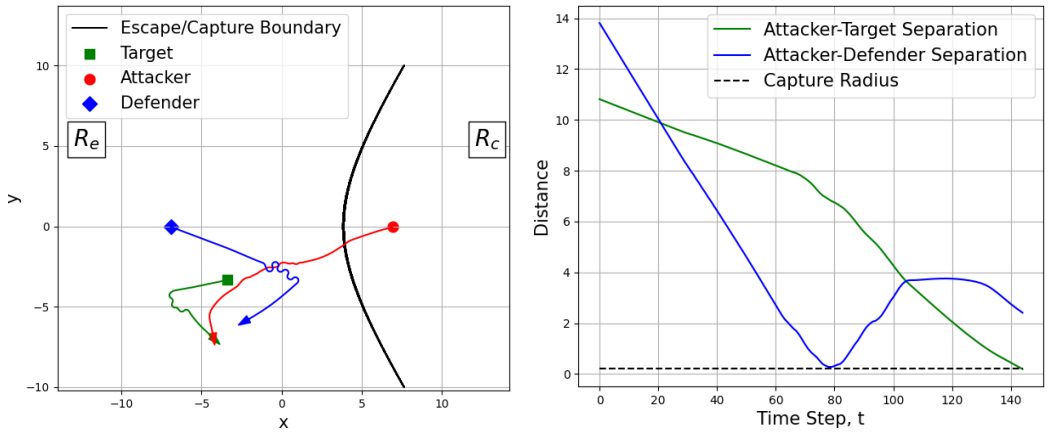


(a) Learned Behavior

(b) Agent separation

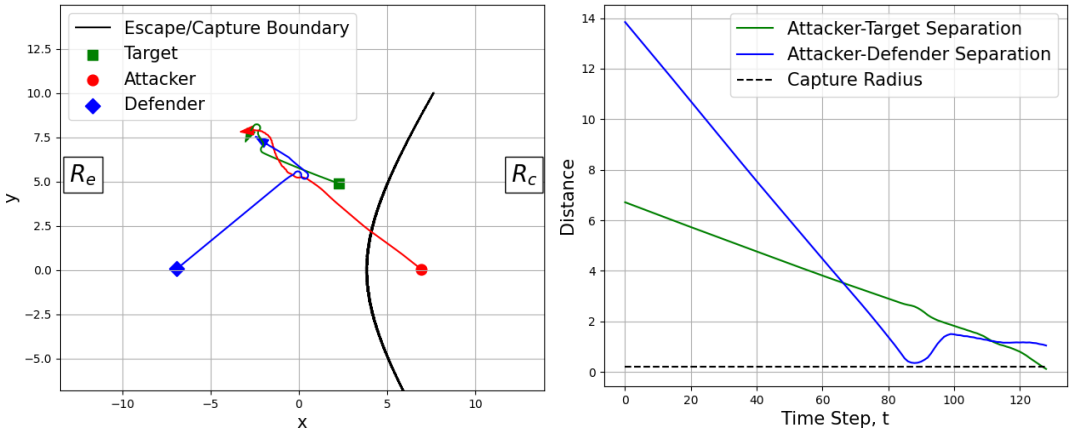
Figure 4.11: Region of Capture $T_0 = (10.00, 5.55)$

In the games starting in the region of escape, defined by the optimal point capture solution, where the trained Attacker succeeded in capturing the Target, the learned policy leveraged the turn-rate limitations of the Defender by forcing it to commit to a heading in pursuit and then causing a change to the optimal defensive strategy output that the Defender is incapable of achieving. This behavior allows the Attacker to avoid the Defender and go on to capture the Target, whereas the point capture attacking guidance law caused the Attacker to be intercepted before reaching the Target. The learned strategy for cases in R_e can be observed in Figures 4.12, 4.13, and 4.14, where the games played by the defender-aware policy are shown. The behavior of the Attacker in the cases shown is representative of the engagements studied.



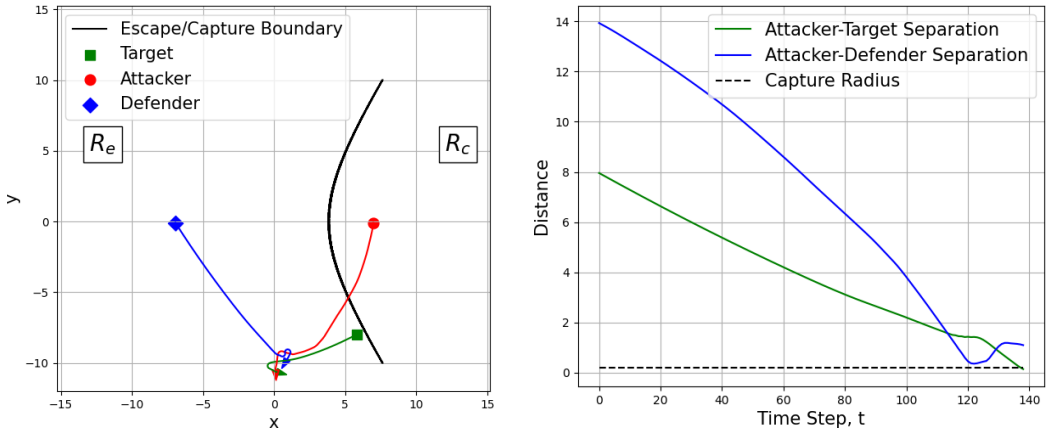
(a) Learned Behavior (b) Agent separation

Figure 4.12: Region of Escape $T_0 = (-3.33, -3.33)$



(a) Learned Behavior (b) Agent separation

Figure 4.13: Region of Escape $T_0 = (2.30, 4.87)$

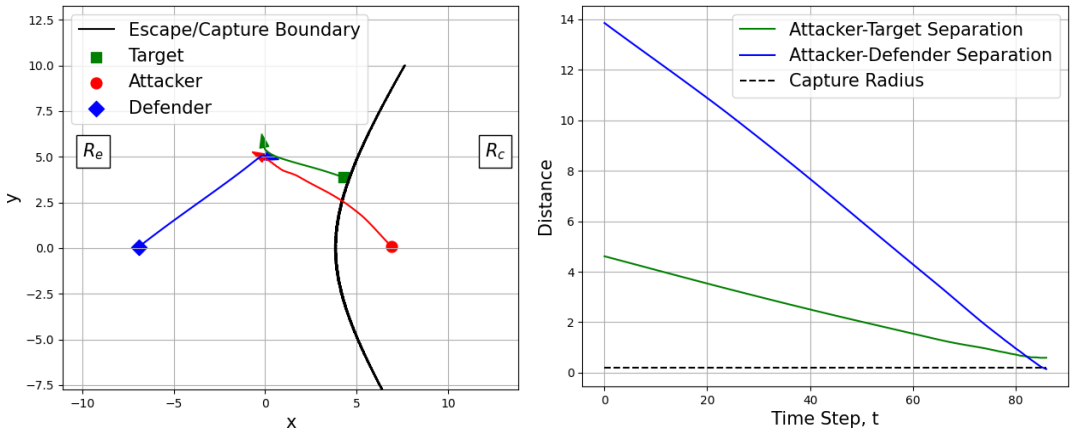


(a) Learned Behavior

(b) Agent separation

Figure 4.14: Region of Escape $T_0 = (5.89, -7.94)$

It is apparent that the Target initializations near the boundary were difficult for the Attacker to learn to win as those are the cases where all three agents will converge to a state in which they are in close proximity. The Attacker's difficulty in capturing the Target due to small errors when the agents converge is shown in Figure 4.15, where the policy failed on a boundary game due to a suboptimal pursuit path.

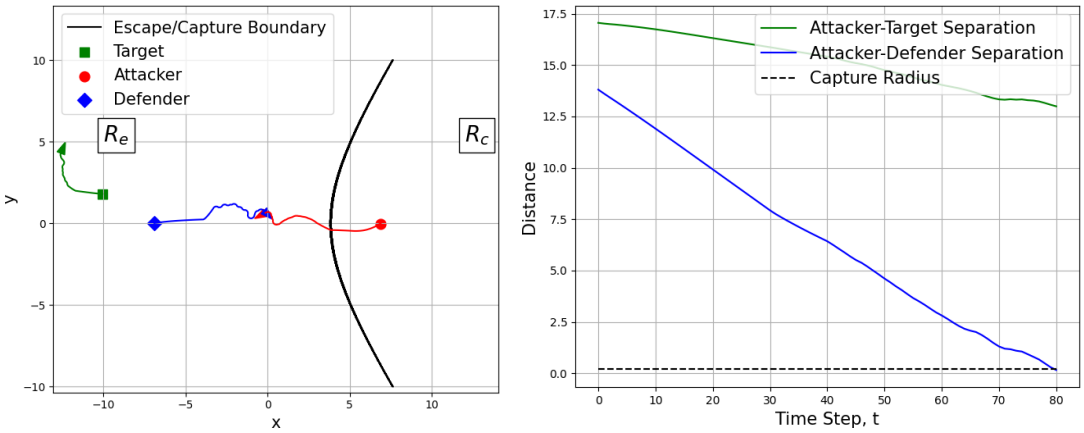


(a) Learned Behavior

(b) Agent separation

Figure 4.15: Boundary Case $T_0 = (4.36, 3.84)$

The simulation environment checked for a violation of the capture radius between ($d_{Capture}$) the Attacker and the Target before the Defender for the terminal game state. Engagements where the convergence can be observed that resulted in a loss for the Attacker were due to the terminal states within Γ_{Loss}. The learned policy also had a lower rate of success for games with initial Target states that caused the Attacker and Defender to approach each other in a near head-on trajectory. In the cases where the near head-on trajectories occurred, the point capture defence was able to match the turns made by the Attacker leading to the Target's survival. The mirroring behavior of the Defender for an example near head-on Attacker loss is shown in Figure 4.16.



(a) Learned Behavior (b) Agent separation

Figure 4.16: Near Head-On Case $T_0 = (-10.00, 1.79)$

The optimal point capture attacking guidance law is compared to the defender-aware guidance policy that resulted through training an agent with TD3 in Table 4.1. The learned policy was able to win a large portion of the R_c cases that the point capture solution was designed for in both the training grid (96.15%) and high-resolution grid (93.62%), but did not achieve complete coverage as discussed above. For the R_e cases, the learned policy showed significant improvement over the point capture guidance, as any win in this region is beyond the previous method's capabilities. The defender-aware attacking guidance policy had a win coverage of 97.29% on the training cases and 86.95% on the high-resolution grid. The number of wins in R_e within the Target initialization bounds gives the learned policy 97% and 88.54% coverage overall of the training and high-resolution cases. Due to the size of R_c versus R_e in the problem space, the point capture guidance law had wins in only 26% (26/100) of the training games and 23.70% (2370/10000) of the larger set evaluated.

Table 4.1: Attacking Guidance Win Coverage

	Training Cases			Testing Cases		
	R_c	R_e	Total	R_c	R_e	Total
Point Capture Guidance Law	100%	0%	26%	100%	0%	23.70%
Defender-Aware Policy	96.15%	97.29%	97%	93.62%	86.95%	88.54%

The minimum distance between the Attacker and Defender in each of the test games provides a different view of performance beyond just a successful target capture. Two areas stand out that reflect the existing optimal guidance boundaries. The known target capture region is first, showing a large distance separation compared to the known Defender inception area. Because games with T_0 positions that are close together cause comparable agent behaviors, the minimum Attacker-Defender separation shown in Figure 4.17 is partitioned into ten value ranges to group similar engagements. The data collected shows that the escape/capture surface B is still present in the turn rate constrained version of the TAD game, where it can be viewed as the limit after which avoiding the Defender is required in the Attacker's pursuit of the Target. An anomaly, located at $(6.0, 0.0)$, exists in the known target capture area due to training attempting to maximize the accumulated reward signal though the learned policy. The Target was ultimately captured in those cases, but the Attacker learned to prolong the games allowing the Defender to decrease its distance to the Attacker. The second region of interest consisted of cases where the minimum distance between the Attacker and Defender was below 0.6. Separating from the known Target capture region, this gives a clearer picture of how successful the learned policy was in avoiding the Defender by selecting all cases beyond the surface B in R_e (Figure 4.18). Games that ended in a win for the Attacker close to games that resulted in a loss generally were engagements where the Attacker's avoidance behavior caused it to

come in close proximity to the Defender, while areas of the T_0 space that had a high density of Attacker wins contained cases where the Attacker-Defender separation was consistently higher. These two regions were separated graphically such that the Defender to Attacker minimum distance can be visualized on separate scales providing a more in-depth view beyond a threshold distance that determined a win or loss.

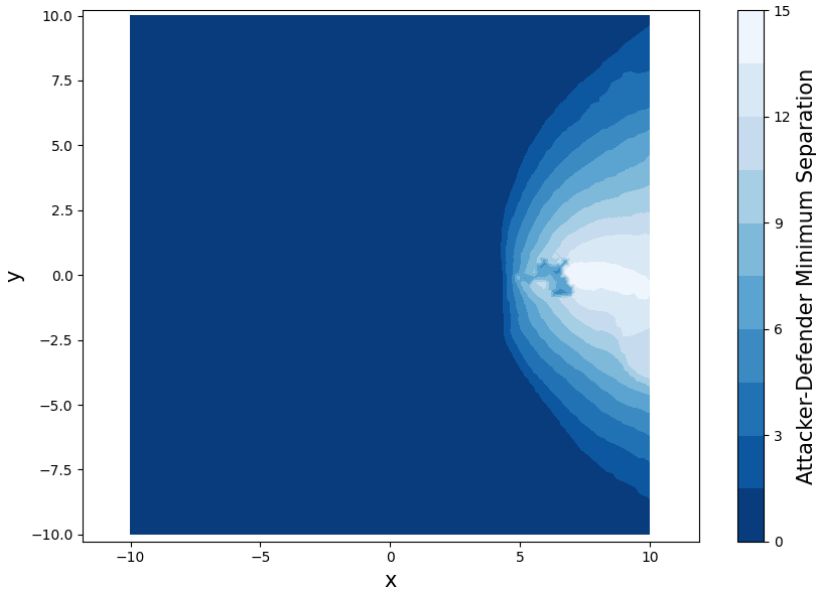


Figure 4.17: Minimum Attacker-Defender Separation in Test Cases

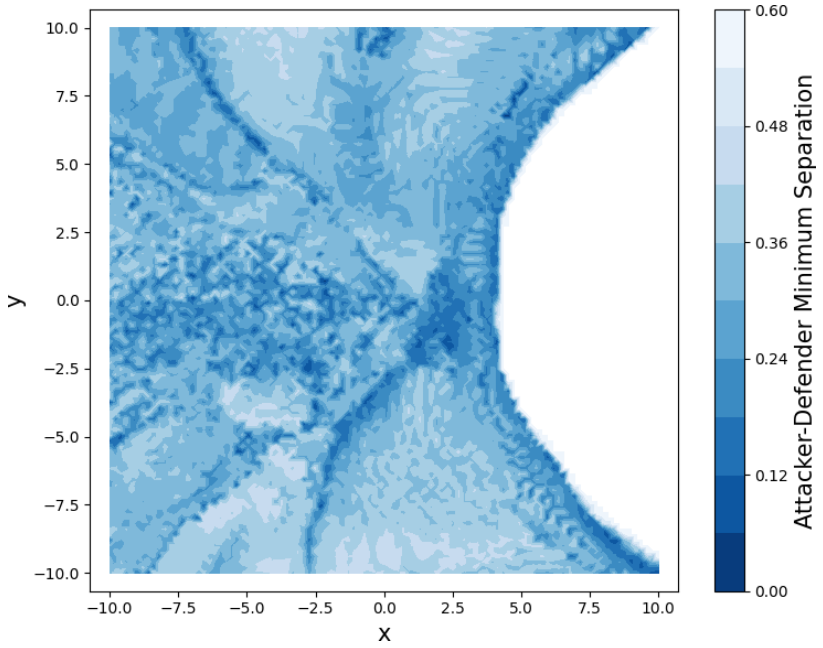


Figure 4.18: Minimum Attacker-Defender Separation in Test Cases ($d_{AT} < 0.6$)

4.5 Summary of Results

The reward function was validated within the developed TAD MDP on a single game belonging to R_e to show that the Attacker could learn a strategy that would allow it to win in a case where the other target pursuit approaches would fail. Hyperparameters were tested and an artificial neural network structure was determined for best performance within the set of Target initial positions. The final policy was produced through further training with the configuration selected. Attacker behaviors from point capture guidance laws were compared with the learned strategy and the Attacker's success in avoiding the Defender was examined.

5 Conclusions

The MDP developed provides a framework in which an attacking agent can learn to capture a defended target in the TAD differential game with turn rate constraints. Cases in which the traditional attacking guidance methods fail to result in the Target's capture were explored with the learned attacking behavior, where the attacking agent sought to maximize its accumulated reward. The set of engagements tested showed that an attacking guidance method with the capability of avoiding the defender is not limited by the previously defined decision barrier. Artificial intelligence was used to expand the approach to attacking guidance, where actions were selected that did not align with the optimal instantaneous heading in order to reach the Target. The defender-aware guidance policy was shown to have a sizable leap in win coverage when compared with the point capture attacking guidance law, increasing the percentage of wins from 26% to 88% within the problem space studied.

Further research into the constrained TAD game could include a range of velocity ratio values and initial positions for the Attacker and Defender in efforts to provide a more generalized attacking guidance policy. Given the guaranteed performance of the optimal solution for target capture in R_c, a fusion of point capture guidance law with the reinforcement learning based guidance policy, that is conditional to the state of the game, would result in a highly effective attacker in TAD scenarios. It is also possible that deep reinforcement learning could provide a method for developing a more robust defensive guidance strategy if the defending team were trained against the defender-aware policy in addition to more conventional attacking guidance laws, in a training approach similar to the method used in [35].

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv:1802.09477 [cs, stat]*, 2018. [Online]. Available: http://arxiv.org/abs/1802.09477

[3] E. Garcia, D. W. Casbeer, K. Pham, and M. Pachter, "Cooperative aircraft defense from an attacking missile," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 2926–2931. [Online]. Available: http://ieeexplore.ieee.org/document/7039839/

[4] Y. Ho, A. Bryson, and S. Baron, "Differential games and optimal pursuit-evasion strategies," *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 385–389, October 1965.

[5] R. L. Boyell, "Defending a moving target against missile or torpedo attack," *IEEE Transactions on Aerospace and Electronic Systems*, no. 4, pp. 522–526, 1976.

[6] R. L. Boyell, "Counterweapon aiming for defense of a moving target," *IEEE Transactions on Aerospace and Electronic Systems*, no. 3, pp. 402–408, 1980.

[7] D. W. Casbeer, E. Garcia, Z. E. Fuchs, and M. Pachter, "Cooperative target defense differential game with a constrained-maneuverable defender," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 1713–1718. [Online]. Available: http://ieeexplore.ieee.org/document/7402457/

[8] E. Garcia, D. W. Casbeer, and M. Pachter, "Pursuit in the presence of a defender," *Dynamic Games and Applications*, vol. 9, no. 3, pp. 652–670, 2019. [Online]. Available: http://link.springer.com/10.1007/s13235-018-0271-9

[9] I. E. Weintraub, R. G. Cobb, W. Baker, and M. Pachter, "Direct methods comparison for the active target defense scenario," in *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, 2020. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2020-0612

[10] M. Pachter, E. Garcia, and D. W. Casbeer, "Active target defense differential game," in *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2014, pp. 46–53. [Online]. Available: http://ieeexplore.ieee.org/document/7028434/

[11] V. Shaferman and T. Shima, "Cooperative multiple-model adaptive guidance for an aircraft defending missile," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 6, pp. 1801–1813, 2010. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.49515

[12] E. Garcia, D. W. Casbeer, and M. Pachter, "Active target defense using first order missile models," *Automatica*, vol. 78, pp. 139–143, 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0005109816305404

[13] A. Ratnoo and T. Shima, "Guidance strategies against defended aerial targets," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1059–1068, 2012. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.56924

[14] T. Shima, "Optimal cooperative pursuit and evasion strategies against a homing missile," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 414–425, 2011. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.51765

[15] M. Pachter, E. Garcia, and D. W. Casbeer, "Toward a solution of the active target defense differential game," *Dynamic Games and Applications*, vol. 9, no. 1, pp. 165–216, 2019. [Online]. Available: http://link.springer.com/10.1007/s13235-018-0250-1

[16] E. Garcia, D. W. Casbeer, and M. Pachter, "Optimal target capture strategies in the target-attacker-defender differential game," in *2018 Annual American Control Conference (ACC)*, June 2018, pp. 68–73.

[17] E. Garcia, D. W. Casbeer, and M. Pachter, "Cooperative strategies for optimal aircraft defense from an attacking missile," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 8, pp. 1510–1520, 2015. [Online]. Available: https://doi.org/10.2514/1.G001083

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: http://www.nature.com/articles/nature14236

[19] Y. Bengio, *Learning Deep Architectures for AI*, ser. Essence of knowledge, The. Now Publishers, 2009. [Online]. Available: https://books.google.com/books?id=cq5ewg7FniMC

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[21] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, "Fast yolo: A fast you only look once system for real-time embedded object detection in video," 2017.

[22] J. Eisenstein, *Introduction to Natural Language Processing*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2019. [Online]. Available: https://books.google.com/books?id=72yuDwAAQBAJ

[23] E. Garcia, D. W. Casbeer, and M. Pachter, "Active target defence differential game: fast defender case," *IET Control Theory & Applications*, vol. 11, no. 17, pp. 2985–2993, 2017.

[24] D. W. Casbeer, E. Garcia, Z. E. Fuchs, and M. Pachter, "Cooperative target defense differential game with a constrained-maneuverable defender," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 1713–1718.

[25] E. Garcia, D. W. Casbeer, Z. E. Fuchs, and M. Pachter, "Aircraft defense differential game with non-zero capture radius," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14 200–14 205, 2017.

[26] R. Isaacs, *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1999.

[27] M. Lau, M. J. Steffens, and D. N. Mavris, "Closed-loop control in active target defense using machine learning," in *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, 2019. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2019-0143

[28] N. A. Shneydor, *Missile guidance and pursuit: kinematics, dynamics and control*. Elsevier, 1998.

[29] L. L. Scharf, W. P. Harthill, and P. H. Moose, "A comparison of expected flight times for intercept and pure pursuit missiles," *IEEE Transactions on Aerospace and Electronic Systems*, no. 4, pp. 672–673, 1969.

[30] M. Guelman, "The closed-form solution of true proportional navigation," *IEEE Transactions on Aerospace and Electronic Systems*, no. 4, pp. 472–482, 1976.

[31] G. M. Siouris, "Tactical missile guidance laws," *Missile Guidance and Control Systems*, pp. 155–267, 2004.

[32] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[33] A. T. Bilgin and E. Kadioglu-Urtis, "An approach to multi-agent pursuit evasion games using reinforcement learning," in *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, 2015, pp. 164–169.

[34] B. Gaudet, R. Furfaro, and R. Linares, "Reinforcement learning for angle-only intercept guidance of maneuvering targets," *arXiv:1906.02113 [cs]*, 2019. [Online]. Available: http://arxiv.org/abs/1906.02113

[35] J. K. Price, O. J. Pinon-Fischer, and D. N. Mavris, "Definition of optimal agent behaviors using reinforcement learning," in *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, 2019. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2019-2200

[36] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[37] I. H. Witten, "An adaptive optimal controller for discrete-time markov environments," *Information and Control*, vol. 34, no. 4, pp. 286–295, 1977. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0019995877903540

[38] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[39] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," p. 9, 2014.

[40] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, 2019. [Online]. Available: http://arxiv.org/abs/1509.02971

[41] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv:1509.06461 [cs]*, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[42] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[43] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.

[44] J. Randlov and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping," *Proceedings of the 15th International Conference on Machine Learning*, pp. 463–471, 01 1998.

[45] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Reward function and initial values: Better choices for accelerated goal-directed reinforcement learning," in *Artificial Neural Networks – ICANN 2006*, S. D. Kollias, A. Stafylopatis, W. Duch, and E. Oja, Eds. Springer Berlin Heidelberg, 2006, vol. 4131, pp. 840–849. [Online]. Available: http://link.springer.com/10.1007/11840817_87

[46] S. M. Kakade *et al.*, "On the sample complexity of reinforcement learning," Ph.D. dissertation, University of London London, England, 2003.

[47] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

APPENDIX

Final Hyperparameter Values and Network Structure for Training

Table A.1: Final Training Configurations

Reward Function	
β_{AT}	$20.6 * 10^{-2}$
σ_{AT}	20.0
β_{AD}	$15.0 * 10^{-2}$
σ_{AD}	0.2
κ	$3.7 * 10^{-2}$
ι	$13.6 * 10^{-2}$
TD3	
Discount Factor, γ	0.99
Target Update Rate, τ	$5 * 10^{-3}$
Exploration Noise	0.2
Batch Size	256
Policy Update Frequency	2
Actor Network Structure	$9 \times 256 \times 512 \times 512 \times 256 \times 1$
Actor Learning Rate	$3 * 10^{-4}$
Critic Network Structure	$10 \times 256 \times 512 \times 512 \times 256 \times 1$
Critic Learning Rate	$3 * 10^{-4}$

Thesis and Dissertation Services