Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

A DATA-DRIVEN DISCRETIZED CS:GO SIMULATION ENVIRONMENT TO FACILITATE STRATEGIC MULTI-AGENT PLANNING RESEARCH

Yunzhe Wang[1,2], Volkan Ustun[1], and Chris McGroarty[3]

[1]USC Institute for Creative Technologies, Los Angeles, CA, USA
[2]Dept. of Computer Science, University of Southern California, Los Angeles, CA, USA
[3]U.S. Army Combat Capabilities Development Command, Orlando, FL, USA

ABSTRACT

Modern simulation environments for complex multi-agent interactions must balance high-fidelity detail with computational efficiency. We present DECOY, a novel multi-agent simulator that abstracts strategic, long-horizon planning in 3D terrains into high-level discretized simulation while preserving low-level environmental fidelity. Using Counter-Strike: Global Offensive (CS:GO) as a testbed, our framework accurately simulates gameplay using only movement decisions as tactical positioning—without explicitly modeling low-level mechanics such as aiming and shooting. Central to our approach is a waypoint system that simplifies and discretizes continuous states and actions, paired with neural predictive and generative models trained on real CS:GO tournament data to reconstruct event outcomes. Extensive evaluations show that replays generated from human data in DECOY closely match those observed in the original game. Our publicly available simulation environment provides a valuable tool for advancing research in strategic multi-agent planning and behavior generation.

1 INTRODUCTION

Team-based multiplayer strategy games have emerged as grand challenge domains for multi-agent learning and long-horizon planning. Breakthroughs in complex games such as StarCraft II and Dota 2—where AI agents have achieved human-expert or even superhuman performance through large-scale self-play training (Baker et al. 2019; Vinyals et al. 2019; Berner et al. 2019; Open Ended Learning Team et al. 2021)—demonstrate that, given sufficient simulation and training, sophisticated strategies can be discovered even in environments characterized by long time horizons, imperfect information, and high-dimensional state-action spaces. However, these successes come at the expense of enormous computational costs, and a prevailing trend in the field is to scale performance by increasing model size and training on larger datasets with more simulation steps (Neumann and Gros 2022; Obando-Ceron et al. 2024; Kaplan et al. 2020). In numerous real-world settings—team sports, emergency response, and search and rescue among them—high-fidelity simulators are either unavailable or prohibitively expensive in both computational and financial terms. Furthermore, many existing strategy game environments, like StarCraft and Dota, utilize isometric or "pseudo-3D" perspectives rather than authentic three-dimensional worlds with first-person viewpoints that more accurately represent real-world scenarios.

Recent advances in offline, model-based, and in-context (reinforcement) learning—such as world models—offer promising paths to address these challenges. Offline RL improves sample efficiency by learning from fixed datasets, while mitigating issues like distributional shift and overestimation bias, and has shown success in both game and robotics domains (Reed et al. 2022; Mathieu et al. 2023; O'Neill et al. 2024; Laskin et al. 2022; Nikulin et al. 2024). Similarly, world models have enabled effective planning in compact latent spaces, achieving strong performance in single-agent tasks (Ha and Schmidhuber 2018; Hafner et al. 2023; Li et al. 2024; Garrido et al. 2024). Yet, their potential for team-based strategic decision-making scenarios remains relatively unexplored.

In parallel, real-world sports analytics—especially in soccer—has driven the development of tactical AI models that learn to predict and generate team behaviors and strategies from player trajectories and game events (Tuyls et al. 2021; Omidshafiei et al. 2022; Wang et al. 2024). However, sports data often remains inaccessible due to privacy concerns and proprietary restrictions (Socolow and Jolly 2017). In contrast, the electronic sports (e-sports) domain provides abundant, high-fidelity data that is more amenable to large-scale modeling and experimentation (Xenopoulos and Silva 2022; Xenopoulos 2023). Counter-Strike: Global Offensive (CS:GO), a popular 5v5 first-person shooter, offers such rich strategic complexity—encompassing coordinated team tactics, resource management, and decision-making under partial observability. In each match, two teams—the Terrorists (T) and Counter-Terrorists (CT)—compete in a series of objective-based rounds, where the Terrorists aim to plant a bomb at a designated site, and the Counter-Terrorists must prevent the plant or defuse the bomb once planted. Success depends not only on mechanical skill but also on timing, positioning, communication, and resource management. Recent work has demonstrated that agents trained directly on human gameplay logs in CS:GO can replicate expert-level movement (Durst et al. 2024). However, a fast, learning-compatible simulator for the game remains unavailable, despite its potential for enabling sample-efficient training and large-scale evaluation of AI policies in high-dimensional, multi-agent strategic decision-making settings.

In this paper, we introduce DECOY, a novel simulation environment designed to support research in strategic multi-agent planning using CS:GO as a testbed. DECOY aligns high-fidelity human gameplay data into an abstracted, discretized simulation framework that enables efficient modeling of complex team behaviors and long-horizon tactical decisions. Without explicitly modeling low-level mechanics such as aiming, recoil, or animation, DECOY focuses on strategic-level decision-making. It leverages a waypoint-based navigation system for action abstraction, along with predictive and generative models trained with real CS:GO tournament data to estimate action outcomes such as shooting damage and engagement results. This abstraction enables fast simulation and reduced action complexity while preserving the original game data distribution, offering a practical and scalable platform for advancing multi-agent learning and strategic behavior generation in e-sports and beyond. Our simulator and pre-trained models are publicly available at github.com/HATS-ICT/decoy.

2 THE DECOY SIMULATOR

We present **DECOY** (**D**iscrete-**E**vent **CO**unter-Strike simul**Y**tor), a Python-based, multi-agent learning simulation environment to support strategic planning in 3D terrain in the game CS:GO. DECOY provides a Gym-style Application Programming Interface (API) to benchmark standard Multi-Agent Reinforcement Learning (MARL) algorithms and is calibrated with human-trajectory tournament data for offline and model-based learning. The framework (Fig 1) comprises three core components: (1) a physics-based 3D environment, (2) a waypoint-based discretization system for movement abstraction, and (3) neural predictive and generative models for event outcome prediction and generation. We introduce the framework in this section and evaluate its performance and alignment with the original game in section 3

2.1 3D Environment

The simulation environment is built with Panda3D (Carnegie Mellon Entertainment Technology Center and Walt Disney Imagineering 2024) and the Bullet Physics library for agent control and collision detection. Environment stepping, game states, and actions follow the PettingZoo API (Terry et al. 2021), ensuring compatibility with standard MARL algorithms. While CS:GO is typically 5v5, our setup supports learning between any number of agents.

To replicate accurate 3D map and physics, we decompiled the `de_dust2` map using online tool BSP-Source (Community 2025), retaining only geometry and removing decorative assets. Agent physics—dimensions, speed, and jumping—follow CS:GO specifications, with Hammer Units from Source Engine converted to meters. The simulation proceeds as discrete events with incremental time steps, with actions guided by

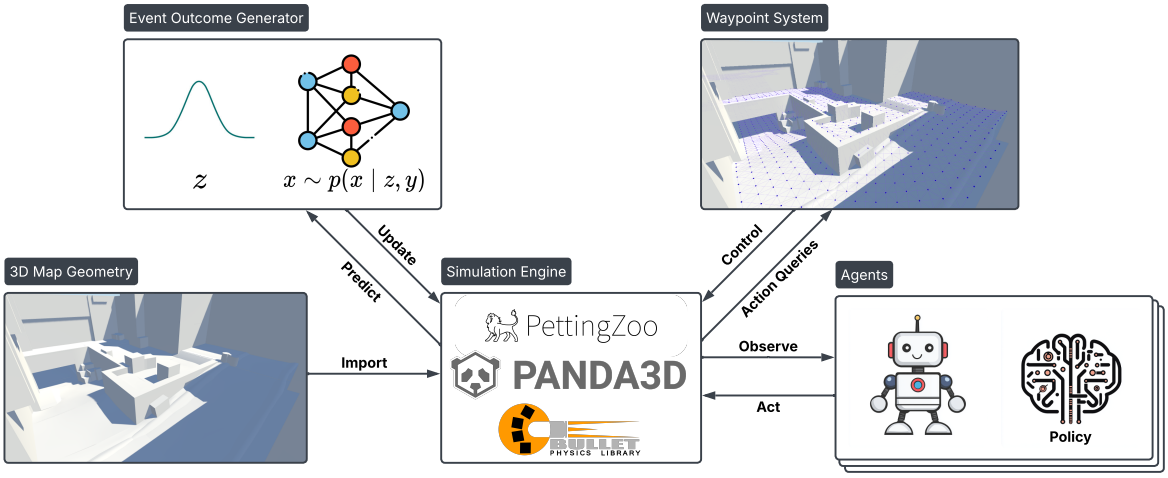

Figure 1: Overview of the DECOY simulation framework. A 3D CS:GO map, Dust II (de_dust2) is imported into the simulation engine and discretized by generated waypoints. Given agent game states, pre-trained predictive and generative models reconstruct outcomes of low-level interactions, such as shooting and damage.

a waypoint structure (Section 2.2). The state space includes data for 10 agents—position, view angle, weapon, armor, helmet—and the bomb's position/state. The action space covers movement, view direction, and stopping.

2.2 Waypoint Discretization

We adopted the same waypoint control setup as Koresh et al. (2024), where prior work has demonstrated its effectiveness in aligning movement trajectories (Ustun et al. 2024). The waypoint graph is a directed graph where vertices denote specific 3D location, and edges represent feasible direct movements between positions. The graph forms a lattice over the map, with fixed distances between waypoints. Each edge is labeled with one of the eight compass directions (N, NE, E, SE, S, SW, W, NW).

Because map navigation involves elevation changes (e.g., ramps), jumping onto boxes, and the fact that diagonal directions cover more distance than cardinal ones, agents may require different numbers of frames to complete the eight directional actions. To address this, agents operate in an on-demand fashion: a new action is requested after the agent reaches its current waypoint, and agent actions proceed asynchronously.

We extended the waypoint generation algorithm of Koresh et al. (2024) to handle complex 3D environments with varying elevation and multi-floor layouts using a three-stage process:

- **Breadth-First Search (BFS) Generation:** Starting from manually specified coordinates, we use a BFS style algorithm to lay waypoints as a bidirectional graph across the 3D map. Various ray tests find neighboring points at proper elevations and avoid placing waypoints inside walls.
- **Drop-Floor Check:** In a second pass, we iterate over all generated waypoints and add singly directed edges from higher to lower elevations that were not previously connected. This enables agents to drop from higher floors to lower ones.
- **Waypoint Verification:** In the final pass, we iterate over all directed edges by placing actual agents to simulate movement between endpoints. This process eliminates edge cases, such as obstacles that block agent movement but were missed in earlier checks. Any blocked edges are removed. The resulting waypoint graph is guaranteed to have no dead ends and to be strongly connected.

In rare cases where critical areas are not covered by the waypoint generation algorithm (e.g., the narrow jumping edge on the X-Box), we manually place waypoints.

2.3 Human Trajectory Dataset

Although the environment is abstracted and discretized, we calibrate its dynamics and data distribution using real human gameplay data. This is achieved by training predictive and generative models (see Section 2.4) and evaluating environment fidelity through replaying original human gameplay trajectories (see Section 3). Specifically, we use the Esports Trajectories & Actions (ESTA) dataset (Xenopoulos and Silva 2022), a comprehensive collection of professional CS:GO match data. This dataset captures complete player trajectories throughout game rounds, including player states, shooting events, damage instances, bomb status, and game outcomes.

We preprocess the dataset by treating each round independently and extracting relevant information for both replay and model training. The extraction process involves parsing official match recordings from professional tournaments to build a structured representation of player movements and interactions.

For each player $i \in \{1, 2, \ldots, 10\}$ at time step t, we extract a state vector

$$s_i^t = (p_i^t, v_i^t, h_i^t, e_i^t)$$

where $p_i^t = (x_i^t, y_i^t, z_i^t) \in \mathbb{R}^3$ is the position, $v_i^t \in [0, 360)$ is the view angle, $h_i^t \in [0, 100]$ is the health, and e_i^t encodes the equipment status (including weapon type, armor value, and helmet presence).

A player's trajectory over a complete round is represented as a sequence $\tau_i = (s_i^1, s_i^2, \ldots, s_i^T)$, where T is the total number of time steps in the round. These trajectories are synchronized to a common timeline, allowing us to reconstruct the complete game state at any moment as $S^t = (s_1^t, s_2^t, \ldots, s_{10}^t)$.

We also extract all damage events as tuples (a, v, d, g, t), where a is the attacker index, v is the victim index, $d \in \mathbb{R}^+$ is the damage amount, $g \in \mathcal{G}$ is the hit group $\mathcal{G} = \{\text{Head}, \text{Neck}, \text{Chest}, \text{Stomach}, \text{Arm}, \text{Leg}\}$, and t is the timestamp. Since multiple shots may occur between movement updates, we align damage events with the nearest movement trajectory time tick and aggregate damage by summing over aligned events.

The processed dataset contains approximately 1.5k matches, comprising over 41k rounds and 8 million frames across 8 maps. Each round contains up to 155 seconds of gameplay, sampled at 2 Frames Per Second (FPS). We split the dataset into training, validation, and test sets using an 8:1:1 ratio. All available data is used for training, while inference and replay evaluations are conducted exclusively on the Dust II map.

2.4 Damage Event Models

A key challenge in our simulation environment is accurately modeling combat outcomes without requiring agents to control precise aiming mechanics. To address this, we developed two probabilistic models that predict engagement outcomes: (1) a **Damage Indicator Predictor (DIP)**, which determines whether any damage is likely to occur in a given tactical situation; and (2) a **Damage Outcome Generator (DOG)**, which if damages is predicted, produces a detailed combat outcomes such as damage amount, death events, and hit locations, based on the learned data distribution.

For simplicity, we model damage events as interactions between agent pairs—specifically, one attacker and one victim. During each inference step, a batch of $5 \times 5 \times 2 = 50$ agent pairs (covering all possible attacker-victim combinations across both teams and directions) is processed to assess the complete tactical situation.

We describe the model architectures for DIP and DOG in Section 2.4.1 and Section 2.4.2, respectively, and present model evaluation results in Section 3.2.

2.4.1 Damage Indicator Predictor (DIP)

The Damage Indicator Predictor (DIP) is a binary classifier that predicts whether a damage event will occur between an attacker and a victim based on contextual game state features. At time step t, input features are $x = \{s^t_{\text{attacker}},\ s^t_{\text{victim}},\ x_{\text{mapId}}\}$, including map ID, positions, angles, weapon, and armor status. These are encoded by a shared Game State Encoder $f_{\text{enc}}(\cdot)$, a multi-layer perceptron (MLP), into a latent representation:

$$\mathbf{h}_{\text{cond}} = f_{\text{enc}}(x) \in \mathbb{R}^d$$

The encoded features are passed to a classification head:

$$\hat{y}_{\text{DIP}} = f_{\text{cls}}(\mathbf{h}_{\text{cond}}) \in \mathbb{R}$$

A sigmoid activation is applied during training for binary cross-entropy loss, and thresholded during inference. DIP evaluation is presented in Section 3.2.1.

2.4.2 Damage Outcome Generator (DOG)

If DIP predicts a potential damage event, the input is passed to DOG—a Conditional Variational Auto-Encoder (CVAE) (Kingma and Welling 2013; Sohn et al. 2015)—which generates damage amounts and hit group.

The rationale for using a generative model for damage outcomes, rather than a discriminative one, stems from the nature of damage mechanics in CS:GO. Damage is influenced by multiple factors, most prominently weapon type and hit group. For instance, a headshot with an Avtomat Kalashnikova (AK)-47 typically results in a one-shot kill, while a body shot does not. However, since our simulator does not explicitly implement aiming—and it is non-trivial to implement realistic aiming without producing behavior that resembles an aimbot—hit locations are not directly modeled.

We experimented with a discriminative model that directly predicts damage values, as well as a multitask variant that predicts both damage and hit group using two output heads. Both approaches failed to capture the original data distribution: the former tends to regress toward the statistical mean, while the latter captures only marginal distributions and fails to model the joint dependency between hit group and damage.

Let $\mathbf{h}_{\text{cond}}$ denote the conditional features from the game state encoder, $\mathbf{d} \in \mathbb{R}$ be the ground-truth damage value, and $\mathbf{g} \in \{0,1\}^k$ be the one-hot hit group label, where k is the number of hit groups.

Training Phase. During training, the encoder maps inputs to latent parameters:

$$\mathbf{z}_{\text{in}} = \text{concat}(f_{\text{embed}}(\mathbf{d}), f_{\text{embed}}(\mathbf{g}), \mathbf{h}_{\text{cond}}) \in \mathbb{R}^m$$
$$\mu, \log\sigma^2 = f^{\text{VAE}}_{\text{enc}}(\mathbf{z}_{\text{in}})$$

A latent variable $\mathbf{z}$ is sampled using the reparameterization trick:

$$\mathbf{z} = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

The sampled latent vector and conditional features are passed to the decoder:

$$\mathbf{z}_{\text{dec}} = \text{concat}(\mathbf{z}, \mathbf{h}_{\text{cond}})$$
$$\hat{\mathbf{d}}, \hat{\mathbf{g}} = f^{\text{VAE}}_{\text{dec}}(\mathbf{z}_{\text{dec}})$$

Here, $\hat{\mathbf{d}}$ is the predicted damage value and $\hat{\mathbf{g}} \in \mathbb{R}^k$ are the logits for the hit group classification.

The model is trained using a loss function derived from the Evidence Lower BOund (ELBO), with both supervised reconstruction terms and a Kullback-Leibler (KL) divergence regularization:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}|\mathbf{d},\mathbf{g})}\left[\lambda_d \cdot \|\hat{\mathbf{d}} - \mathbf{d}\|^2 + \lambda_g \cdot \text{CE}(\hat{\mathbf{g}}, \mathbf{g})\right] + \lambda_{\text{KL}} \cdot D_{\text{KL}}\left(q(\mathbf{z} \mid \mathbf{d}, \mathbf{g}) \,\|\, p(\mathbf{z})\right)$$

where $\mathrm{CE}(\cdot)$ denotes the cross-entropy loss over hit groups, and $\lambda_{\mathrm{d}}, \lambda_{\mathrm{g}}, \lambda_{\mathrm{KL}}$ are tunable hyperparameters that control the relative weighting between the regression term, classification loss, and KL regularization.

Inference Phase. During generation, the VAE encoder is removed, and the latent variable $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is sampled directly from the prior and combined with the conditional context to decode sampled damage values and hit group predictions.

We evaluate both the reconstruction mode and the generative mode (sampling from the prior) in Section 3.2.2, and analyze the learned latent space $\mathbf{z}$ in Section 3.2.3.

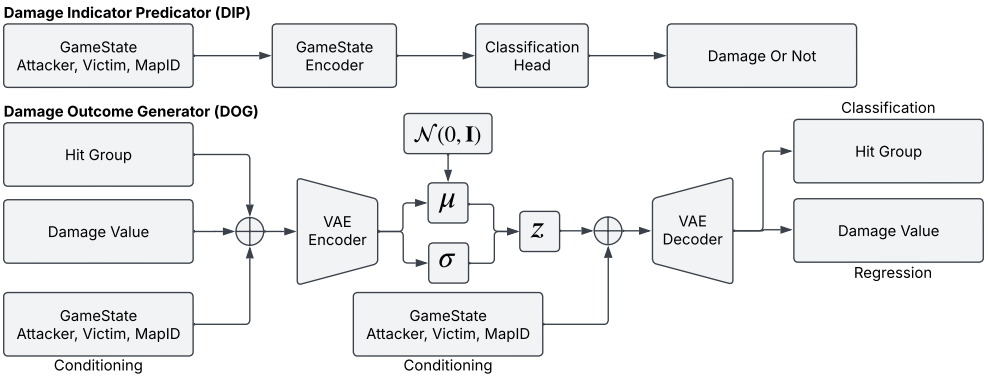

Figure 2: Architecture of the Damage Indicator Predictor (DIP) and Damage Outcome Generator (DOG). A shared Game State Encoder processes context, while a Variational Auto-Encoder (VAE) maps damage and hit group outcomes to a latent space. During generation, only the decoder is used by sampling latent variables from $\mathcal{N}(0, I)$.

3 RESULTS

We evaluate DECOY on three key aspects: (1) its simulation speed, (2) the ability of the damage event models to replicate the original data distribution, and (3) its accuracy in replaying the original game in terms of player movements, damage outcomes, and final game results.

3.1 Simulation Speed

To evaluate the efficiency of the simulator, we conducted a timing benchmark by running the simulation with varying numbers of agents. Each configuration was run until a total of 10,000 agent decision steps had been completed. Graphics rendering was disabled, and the physics engine was stepped at 60 ticks per second to match the original game's physics fidelity.

The results are summarized in Table 1. Our simulator achieved approximately a 16x speed-up (966 Ticks/sec) over real-time performance in the standard 5v5 (10-agent) scenario. However, a noticeable slowdown was observed as the number of agents increased, due to the increased computational load in the physics simulation.

Table 1: Simulation speed under varying agent counts.

# Agents	# Physics Ticks	# Decisions	Wall Time (s)	Physics Time (s)	Ticks/sec	Time Scale
2	47,373	10,000	10.96	789.55	4322.35	72.03
6	15,897	10,000	10.07	264.95	1578.56	26.32
10	9,472	10,000	9.80	157.87	966.53	16.11
20	4,742	10,000	9.73	79.03	487.25	8.12
100	944	10,000	9.48	15.73	99.58	1.66

3.2 Model Evaluation

We evaluate the damage models in terms of classification accuracy, generative fidelity, and representational structure. Specifically, we assess: (1) the ability of the Damage Indicator Predictor (DIP) to detect whether a damage event occurs; (2) the Damage Outcome Generator's (DOG) capability to reconstruct and generate realistic damage values and hit group distributions; and (3) the structure of the learned latent space in the DOG model.

3.2.1 Damage Indicator Model Evaluation

The Damage Indicator Predictor (DIP) achieved an accuracy of 90.8%, an F1-score of 0.913, precision of 87.7%, recall of 94.8%, an Area Under the Receiver Operating Characteristic Curve (AUC-ROC) of 0.951, and an Average Precision (AP) of 0.927 on the holdout test dataset. The high accuracy and F1-score indicate strong overall classification performance. The precision and recall scores highlight the model's effectiveness in correctly identifying damaged cases while limiting false positives. The high AUC demonstrates the DIP's strong capability to discriminate between damaged and undamaged instances. A decision threshold of 0.444 was selected to balance precision and recall optimally, based on precision-recall trade-off analysis.

3.2.2 Damage Outcome Generator Evaluation

The Damage Outcome Generator is evaluated under both reconstruction and generative settings.

In reconstruction mode, the model achieves a Mean Absolute Error (MAE) of 13.97 HP (13.97% of the total damage scale) and an R^2 score of 0.687. The hit group classification accuracy reaches 96.6%. Both results indicate effective reconstruction of damage outcomes. We attribute the remaining prediction errors primarily to the lack of temporal context and the assumption of independent pairwise interactions (see Discussion).

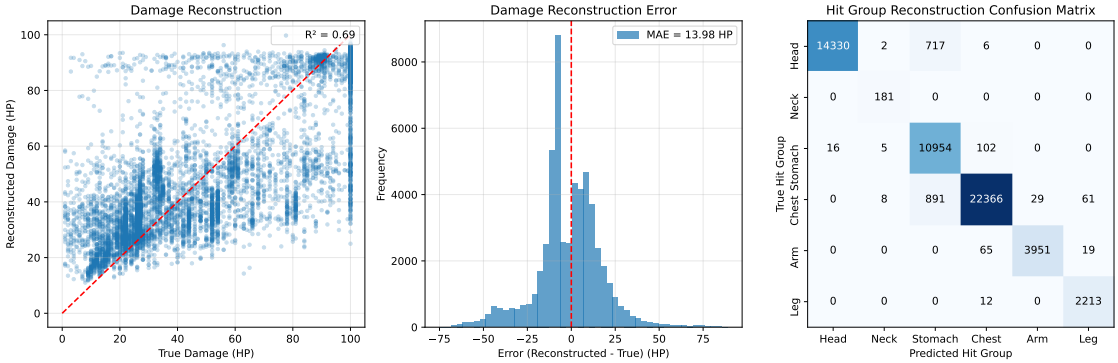

Figure 3: DOG reconstruction evaluation. Left: Actual vs. reconstructed damage (R^2 = 0.687). Center: Reconstruction error histogram. Right: Hit group confusion matrix (accuracy = 96.6%).

In generative mode, the model's performance is assessed through distributional alignment metrics. Figure 4 compares the generated and actual damage distributions across weapon types and hit locations, with mean Wasserstein Distance (WD) 9.87 HP (9.87% of the total damage scale), and confirming that the model generates statistically accurate outcomes.

3.2.3 Latent Space Analysis

We analyze the latent space learned by the DOG model using Unified Manifold Approximation and Projection (UMAP) (McInnes et al. 2018) to project latent vectors into two dimensions. Figure 5 shows the projection colored by damage values (left) and hit groups (right). The damage view reveals smooth gradients from

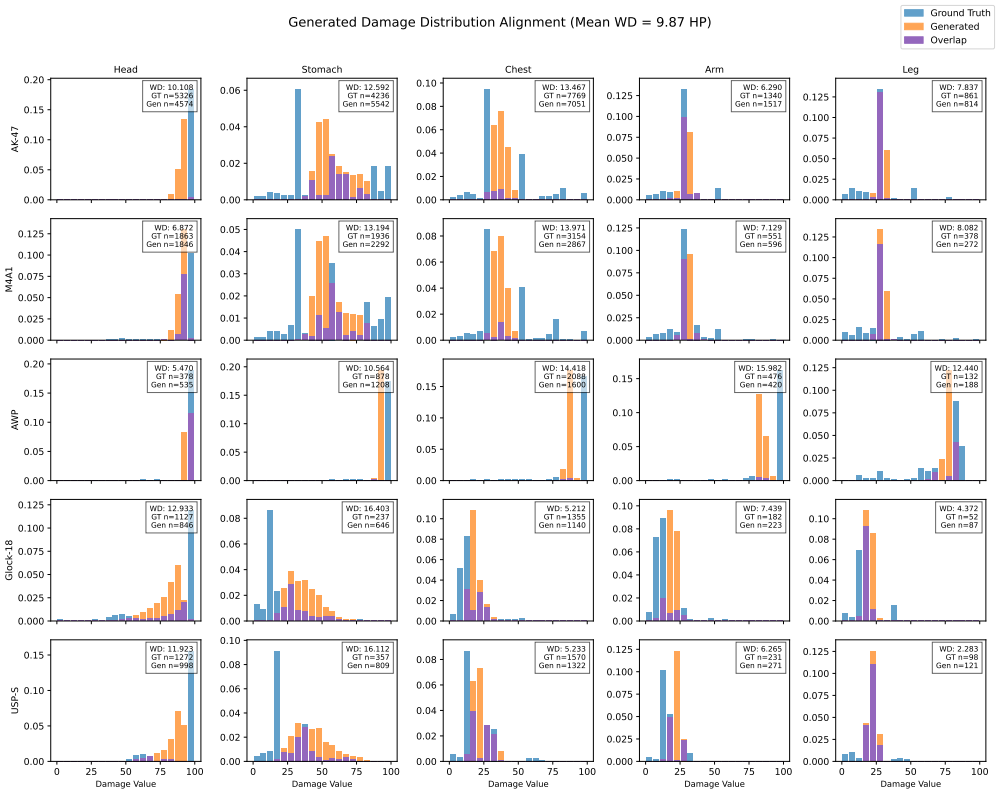

Figure 4: Distributional alignment between generated and true damage patterns across the 5 most common weapons and hit groups. The mean Wasserstein Distance (WD) between predicted and true distributions is 9.87 Health Point (HP)s (9.87% of the total damage scale).

low to high values, while the hit group view shows clearly separated clusters for different body parts. This indicates that the representation jointly captures a continuous encoding of damage severity and a categorical encoding of hit group.

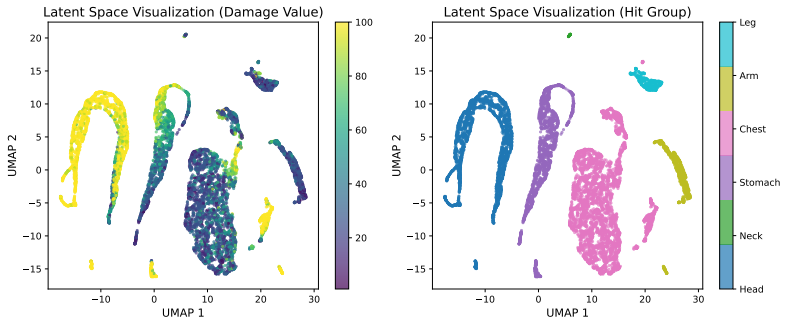

Figure 5: UMAP plot of DOG's latent space. Left: Colored by damage value. Right: Colored by hit-group.

3.3 Human Trajectory Replay

To assess the realism of movement in our simulation environment, we evaluate how well DECOY replicates human player behavior by replaying professional CS:GO trajectories within the simulator.

We begin by converting movement data from the ESTA dataset into waypoint trajectories by mapping each timestamp to its nearest waypoint. If a step skips multiple waypoints, we interpolate the missing

points using shortest path algorithms. This results in a waypoint trajectory that can be translated into a discrete action sequence using only the eight compass directions. The resulting sequence is replayed in the simulator with all game mechanics enabled, including shooting and damage via DIP-DOG, bomb interactions, and final outcomes.

To compare the replayed trajectories with the original human data, we sample simulator outputs at the same frequency as the original trajectory (2 FPS) and normalize all trajectories to the maximum sequence length via interpolation. We then evaluate the spatial and temporal similarity between the original and waypoint trajectories using various metrics, alongside comparisons of health point distribution over time and final game outcomes.

3.3.1 Spatiotemporal Alignment of Movement Trajectories

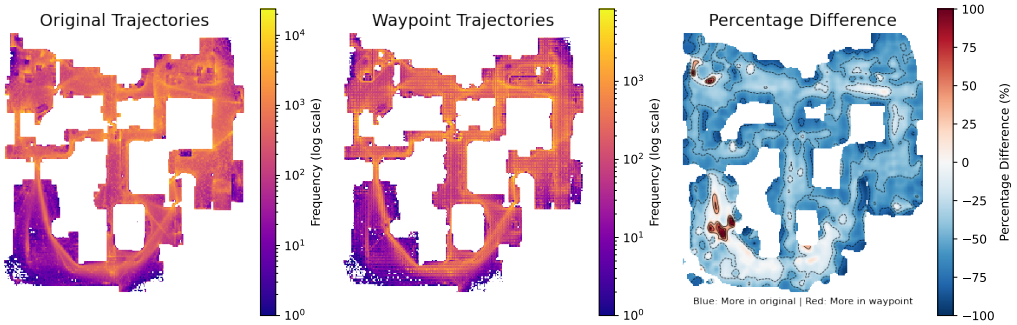

Figure 6: Distribution comparison between the original player trajectory and the replayed waypoint trajectory. Left: Heatmap distribution of original trajectories — brighter colors indicate more frequently visited routes. Middle: Heatmap of waypoint trajectories. Right: Percentage difference contour plots.

We evaluate trajectory fidelity using a set of geometric and temporal similarity metrics: Dynamic Time Warping (DTW) (Salvador and Chan 2007), Euclidean distance, Root Mean Squared Error (RMSE), and Fréchet distance. To isolate the effect of external factors such as damage and player death, we evaluate movement trajectories with shooting and bomb actions disabled. Each metric is computed per player across all rounds and aggregated by team (T vs. CT) as well as overall, with results reported in Table 2.

Table 2: Trajectory alignment metrics comparing replayed (waypoint) and original human trajectories. Values are reported as mean ± standard deviation. Lower is better.

Metric	**T Side**	**CT Side**	**Overall**
DTW	0.873 ± 1.868	1.332 ± 2.263	1.103 ± 2.088
Euclidean	0.420 ± 0.237	0.467 ± 0.252	0.443 ± 0.246
RMSE	5.155 ± 2.911	5.720 ± 3.091	5.437 ± 3.016
Fréchet 2D	4.030 ± 8.599	6.690 ± 9.910	5.360 ± 9.372

The waypoint-based trajectories generally align with the original human trajectories with high spatial and temporal fidelity. Notably, alignment is tighter on the T side than the CT side, possibly because bombsites A and B—closer to the CT spawn—are more cluttered with obstacles, making accurate replay more difficult. The average Euclidean distance is 0.443 meters (with waypoint spacing at 0.7 meters), meaning agents stay within about two-thirds of a waypoint from the intended path, indicating strong spatial accuracy. The higher DTW value compared to Euclidean distance suggests minor temporal misalignments, though the overall trajectory shapes remain preserved. The elevated Fréchet distance (5.36) and RMSE indicate occasional detours, likely due to limited waypoint coverage in less-traveled areas.

We further visualize both the original and waypoint trajectories as heatmaps in Figure 6. Brighter regions indicate more commonly visited paths, and the heatmaps show strong overall alignment between the two. Due to discretization, the waypoint trajectory appears pixelated. We also plot the percentage difference between the two as a contour map, revealing that misalignments tend to occur near walls and obstacles—especially in areas requiring jumps. These deviations may result from agents accidentally falling off ledges or taking unintended detours.

3.3.2 Health Points and Game Results

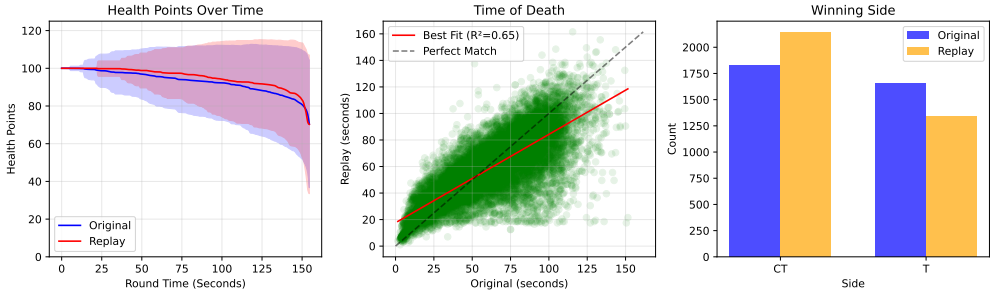

Figure 7: Human replay results. Left: Health point distribution over time (mean ± std). Middle: Correlation between time of death and other factors. Right: Final game outcome distribution.

To evaluate how accurately DECOY reproduces gameplay outcomes, we analyze health point trajectories, death timings, and final match results. A combination of damage event models and line-of-sight detection is used to simulate realistic combat interactions. Bomb planting and defusing are automated: the bomb is planted when the carrier reaches the bombsite, and defusing is triggered when a CT agent reaches the bomb. As shown in Figure 7, DECOY achieves strong alignment with the original game in terms of health progression (HP Correlation: 0.961; HP RMSE: 2.35) and reasonable agreement in death timings (Death Time Correlation: 0.809; MAE: 12.8 seconds). The final match outcome aligns with the original 91.0% of the time. However, the underlying reasons for team victories can differ, largely due to the current DIP-DOG model's assumptions—specifically, the lack of temporal dependency modeling and the use of pairwise conditional independence. These factors further compound to inaccurate bombing actions (see Discussion), where future work is needed to improve fidelity.

4 DISCUSSION

While DECOY provides a fast and tactically grounded simulation for modeling team-based combat in CS:GO at a strategic level, accurately predicting and generating every aspect of the game using only movement remains challenging. A notable issue is the compounding error effect involving movement, shooting damage, and final game outcomes. Movement replay errors can cause agents to drift from their intended positions and timings, which leads to incorrect conflict timings and inaccurate damage calculations. These inaccuracies further cascade into distorted outcomes, particularly around bomb-related actions, as agents may not expire at the correct moments. However, achieving perfect replication of exact CS:GO game outcomes is beyond the scope of this paper.

Several limitations also exist in our current implementation and modeling. DECOY models each round independently, neglecting longer-term factors like team economy and multi-round strategic planning—both essential for realistic agent behavior. For example, players may choose to hide and save weapons for future rounds when at a disadvantage, a tactic not currently represented. Additionally, simulation performance slows as agent count increases. Although physics settings are set to high during replay to match the original game, users can improve speed by lowering physics update frequency and increasing agent speed. The

current simulation also excludes utility mechanics such as grenades, flashbangs, and smoke. Moreover, the DIP and DOG models do not incorporate temporal dependencies, and assume conditional independence in pairwise interactions. This limitation restricts their ability to fully capture the complex joint dynamics and temporal relationships inherent in multi-agent engagements. Empirical evaluation indicates that this lack of temporal dependency often results in premature agent deaths, leading to further inaccuracies in bomb-related actions during replays. Future work will address these limitations by incorporating more expressive temporal modeling methods, such as Graph Neural Networks and Transformers.

Finally, while DECOY is specifically grounded in the structure and mechanics of CS:GO, our broader aim is to support research into tactical and strategic decision-making in real-world scenarios. Recent advances in 3D scene reconstruction, wearable tracking devices, and video-based motion capture technologies increasingly enable realistic environment reconstruction with accurate human movement data. By abstracting such physical environments into navigable waypoint networks and modeling high-level decision events, the simulation framework developed within DECOY holds potential to extend to diverse real-world tactical and strategic domains with the integration of generative models.

5 CONCLUSION

We presented DECOY, a fast, data-driven multi-agent simulation environment inspired by tactical shooter gameplay. By combining waypoint-based navigation with a modular two-stage generative model for damage prediction, DECOY enables efficient and high-fidelity simulation of complex multi-agent engagements. Its alignment with human trajectory data and incorporation of stochastic combat modeling allow it to capture both strategic behavior and the uncertainty inherent in real-world scenarios. DECOY offers a flexible and extensible platform for advancing research in multi-agent tactical decision-making.

ACKNOWLEDGMENTS

The authors acknowledge the use of Large Language Models for assistance with proofreading and grammar checking. All content was reviewed, edited, and approved by the human authors, who take full responsibility for the final manuscript. The project or effort depicted was or is sponsored by the U.S. Army Combat Capabilities Development Command – Soldier Centers under contract number W912CG-24-D-0001. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

REFERENCES

Baker, B., I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew *et al*. 2019. "Emergent Tool Use from Multi-Agent Autocurricula". In *Proceedings of the International Conference on Learning Representations*. New Orleans, Louisiana, United States. May 6–9, 2019.

Berner, C., G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, *et al*. 2019. "Dota 2 with Large Scale Deep Reinforcement Learning". *arXiv preprint arXiv:1912.06680*.

Carnegie Mellon Entertainment Technology Center and Walt Disney Imagineering 2024. "Panda3D". *https://www.panda3d.org*. Version 1.10.14, accessed 23rd March.

Community, V. D. 2025. "BSPSource". *https://developer.valvesoftware.com/wiki/BSPSource*. accessed 23rd March.

Durst, D., F. Xie, V. Sarukkai, B. Shacklett, I. Frosio, C. Tessler, *et al*. 2024. "Learning to Move Like Professional Counter-Strike Players". In *Computer Graphics Forum*, Volume 43, e15173. Wiley Online Library.

Garrido, Q., M. Assran, N. Ballas, A. Bardes, L. Najman, and Y. LeCun. 2024. "Learning and Leveraging World Models in Visual Representation Learning". *arXiv preprint arXiv:2403.00504*.

Ha, D., and J. Schmidhuber. 2018. "World Models". *arXiv preprint arXiv:1803.10122*.

Hafner, D., J. Pasukonis, J. Ba, and T. Lillicrap. 2023. "Mastering Diverse Domains through World Models". *arXiv preprint arXiv:2301.04104*.

Kaplan, J., S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, *et al*. 2020. "Scaling Laws for Neural Language Models". *arXiv preprint arXiv:2001.08361*.

Kingma, D. P., and M. Welling. 2013. "Auto-Encoding Variational Bayes". *arXiv preprint arXiv:1312.6114*.

Koresh, C., V. Ustun, R. Kumar, and T. Aris. 2024, May. "Improving Reinforcement Learning Experiments in Unity through Waypoint Utilization". In *The International FLAIRS Conference Proceedings*, Volume 37.

Laskin, M., L. Wang, J. Oh, E. Parisotto, S. Spencer, R. Steigerwald *et al*. 2022. "In-Context Reinforcement Learning with Algorithm Distillation". *arXiv preprint arXiv:2210.14215*.

Li, J., Q. Wang, Y. Wang, X. Jin, Y. Li, W. Zeng *et al*. 2024. "Open-World Reinforcement Learning over Long Short-Term Imagination". *arXiv preprint arXiv:2410.03618*.

Mathieu, M., S. Ozair, S. Srinivasan, C. Gulcehre, S. Zhang, R. Jiang, *et al*. 2023. "AlphaStar Unplugged: Large-Scale Offline Reinforcement Learning". *arXiv preprint arXiv:2308.03526*.

McInnes, L., J. Healy, N. Saul, and L. Großberger. 2018. "UMAP: Uniform Manifold Approximation and Projection". *Journal of Open Source Software* 3(29):861.

Neumann, O., and C. Gros. 2022. "Scaling Laws for a Multi-Agent Reinforcement Learning Model". *arXiv preprint arXiv:2210.00849*.

Nikulin, A., I. Zisman, A. Zemtsov, V. Sinii, V. Kurenkov, and S. Kolesnikov. 2024. "XLand-100B: A Large-Scale Multi-Task Dataset for In-Context Reinforcement Learning". *arXiv preprint arXiv:2406.08973*.

Obando-Ceron, J., G. Sokar, T. Willi, C. Lyle, J. Farebrother, J. Foerster, *et al*. 2024. "Mixtures of Experts Unlock Parameter Scaling for Deep RL". *arXiv preprint arXiv:2402.08609*.

Omidshafiei, S., D. Hennes, M. Garnelo, Z. Wang, A. Recasens, E. Tarassov, *et al*. 2022. "Multiagent Off-Screen Behavior Prediction in Football". *Scientific Reports* 12(1):8638.

O'Neill, A., A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee *et al*. 2024. "Open X-Embodiment: Robotic Learning Datasets and RT-X Models: Open X-Embodiment Collaboration 0". In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation*, 6892–6903. Yokohama, Japan: IEEE. May 13–17, 2024.

Open Ended Learning Team, A. Stooke, A. Mahajan, C. Barros, C. Deck, J. Bauer *et al*. 2021. "Open-Ended Learning Leads to Generally Capable Agents". *arXiv preprint arXiv:2107.12808*.

Reed, S., K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron *et al*. 2022. "A Generalist Agent". *arXiv preprint arXiv:2205.06175*.

Salvador, S., and P. Chan. 2007. "Toward Accurate Dynamic Time Warping in Linear Time and Space". *Intelligent Data Analysis* 11(5):561–580.

Socolow, B., and I. Jolly. 2017. "Game-Changing Wearable Devices That Collect Athlete Data Raise Data Ownership Issues". *World Sports Advocate* 15(7):15–17.

Sohn, K., H. Lee, and X. Yan. 2015. "Learning Structured Output Representation using Deep Conditional Generative Models". In *Advances in Neural Information Processing Systems*, edited by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Volume 28: Curran Associates, Inc.

Terry, J., B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan *et al*. 2021. "PettingZoo: Gym for Multi-Agent Reinforcement Learning". *Advances in Neural Information Processing Systems* 34:15032–15043.

Tuyls, K., S. Omidshafiei, P. Muller, Z. Wang, J. Connor, D. Hennes, *et al*. 2021. "Game Plan: What AI Can Do for Football, and What Football Can Do for AI". *Journal of Artificial Intelligence Research* 71:41–88.

Ustun, V., S. Hans, R. Kumar, and Y. Wang. 2024. "Abstracting Geo-Specific Terrains to Scale Up Reinforcement Learning". In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) 2024*. Orlando, Florida, United States: National Training and Simulation Association (NTSA).

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, *et al*. 2019. "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning". *Nature* 575(7782):350–354.

Wang, Z., P. Veličković, D. Hennes, N. Tomašev, L. Prince, M. Kaisers, *et al*. 2024. "TacticAI: An AI Assistant for Football Tactics". *Nature Communications* 15(1):1906.

Xenopoulos, P. 2023. *Data Mining for Esports*. Ph. D. thesis, New York University Tandon School of Engineering.

Xenopoulos, P., and C. Silva. 2022. "ESTA: An Esports Trajectory and Action Dataset". *arXiv preprint arXiv:2209.09861*.

AUTHOR BIOGRAPHIES

YUNZHE WANG is a Ph.D. student at the University of Southern California. His research interests include Multi-Agent Learning, Sequential Decision-making, Reinforcement Learning, and Large Language Models. Email: yunzhewa@usc.edu

VOLKAN USTUN is the Associate Director of the Human-Inspired Adaptive Teaming Systems Group at the USC Institute for Creative Technologies. His research augments Multi-agent Reinforcement Learning (MARL) models, drawing inspiration from operations research, human judgment and decision-making, game theory, graph theory, and cognitive architectures to better address the challenges of developing behavior models for synthetic characters, mainly in military training simulations. Email: ustun@ict.usc.edu

CHRIS MCGROARTY is the Chief Engineer for Advanced Simulation at the US Army Combat Capabilities Development Command, Soldier Center, Simulation and Training Technology Center (DEVCOM SC STTC). His research interests include distributed simulation, simulation architectures, applications of Artificial Intelligence Technologies to simulation, novel computing architectures, innovative methods for user-simulation interaction, methodologies for making simulation more accessible by non-simulation experts, future simulation frameworks, and the application of videogame industry technologies. Email: christopher.j.mcgroarty.civ@army.mil