Engagement-Aware Agentic Pursuit-Evasion

Ananya Acharya[1], Trenton Goyette[1], Masoud Ataei[2], Adrian Stoica[1], Vikas Dhiman[2], Mohammad Javad Khojasteh[1]

[1]RIT, Rochester, NY
[2]University of Maine, Orono, ME
{aa2334, twg8622, mjkeme, adseee}@rit.edu, {masoud.ataei, vikas.dhiman}@maine.edu

Abstract

This paper presents a hierarchical multi-agent architecture in which independent large language model (LLM) planners perform strategic role assignment for attacking and defending robot teams, decoupled from low-level control execution. At each planning cycle, each team's LLM planner observes its own team in full but the opposing team only within its robots' combined field of view, then assigns each robot a tactical role – e.g., hold a perimeter, neutralize an intruder on contact, or converge with teammates for capture – together with a natural-language justification. Each robot independently executes its assigned role through a receding-horizon model predictive control (MPC) controller, followed by a discrete-time control barrier function (CBF) filter for safety and role-dependent engagement constraints. Differentiated capture and neutralization incentives require the defending planner to balance threat resolution against resource allocation under partial observability. We evaluate the framework across variable-sized adversary teams using both state-based tactical reasoning and image-based contact classification. Results show that collective team behavior can be adapted through high-level LLM role assignment while retaining the same underlying low-level control architecture. [1]

1 Introduction

A growing body of work integrates foundation models – large language and vision-language models pretrained on broad, task-agnostic data – into robot planning and decision-making. Rather than training policies end-to-end for a single task, these approaches use foundation models to reason over the current task state in natural language and make high-level decisions in a zero-shot manner. (Hu et al. 2023; Firoozi et al. 2025). This shift is particularly relevant to planning specifically: rather than treating a foundation model as a low-level policy to be fine-tuned (Jülg, Burgard, and Walter 2025), a language model can instead sit above a conventional control stack as a strategic layer, issuing structured, human-interpretable decisions that a separate, purpose-built controller then executes (Navaratnarajah et al. 2026). This work situates that idea in an adversarial, multi-agent setting – pursuit-evasion – to test whether large language models (LLMs) can serve as strategic planners under partial observability and competing objectives.

Pursuit-evasion problems provide a general framework for mathematically formalizing a wide range of applications, including surveillance, navigation, and conflict operations. The formulation considers multiple agents organized into two opposing teams: pursuers and evaders. The primary objective is to develop strategies that enable an autonomous agent to act effectively against its opponent (Shishika and Kumar 2020; Weintraub, Pachter, and Garcia 2020; Chung, Hollinger, and Isler 2011; Weintraub, Von Moll, and Pachter 2023).

We formulate pursuit-evasion as a finite-horizon dynamic game between a team of pursuers and a team of evaders, each team acting to optimize its own objective – capture or interception for the pursuers, breach or escape for the evaders – against the other's best response. As team size grows, solving multi-agent pursuit-evasion as a single joint optimization becomes increasingly expensive. We therefore adopt a hierarchical architecture (Matni, Ames, and Doyle 2024) that separates strategic coordination from low-level control. Each robot independently solves a role-conditioned receding-horizon control problem and safety filter, avoiding a joint team-wide control solve. A slower strategic layer determines what objective each controller should pursue – e.g., whether an agent should defend, engage a particular opponent, or reposition. In this first study, this strategic layer is centralized within each team: a single LLM planner assigns roles and targets to all active agents, while control execution remains per-agent. Within this hierarchy, the two teams retain fixed opposing missions – one defends a region and the other attempts to breach it – while two independent LLM planners (one per team) adapt their agents' tactical roles to the evolving tactical picture.

The two teams share the same underlying control architecture, but their tactical vocabularies are different: the defending team chooses among four roles (DEFEND, NEUTRALIZE, CAPTURE, RECON), reflecting its richer set of ways to intercept or contain a threat, while the attacking team chooses between only two (ATTACK, EVADE), reflecting its narrower, single-minded objective of breaching the region while avoiding capture. Within each team's mission, an agent's tactical assignment (e.g. hold position, pur-

[1]Code available at: https://github.com/ananya-ac/llm-pursuit-evasion.git.

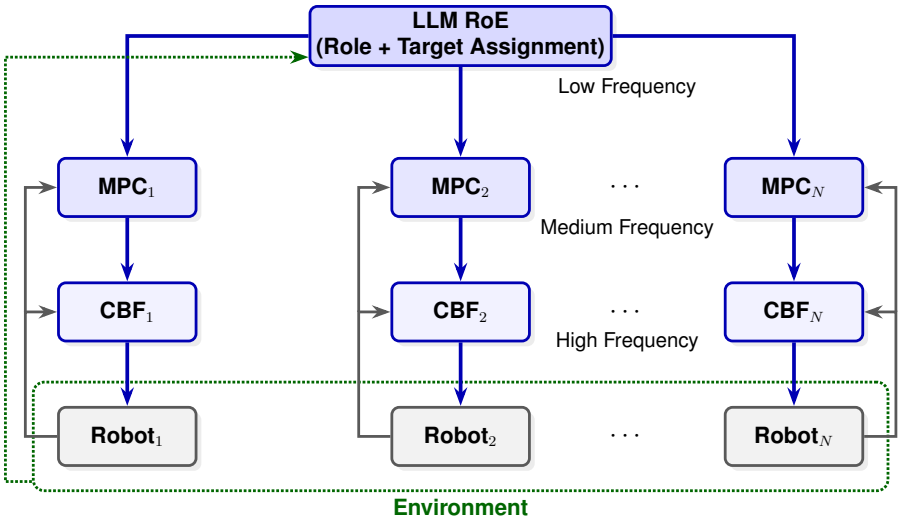


Figure 1: Engagement-Aware Agentic Pursuit-Evasion

sue a specific threat, or scout) is treated as a control-design object rather than the direct output of a game-theoretic solve. This assignment is issued once per planning cycle for both teams, by an LLM planner explicitly over the tactical picture – the LLM's assignment accompanied by a natural-language justification for every agent. At the execution layer, each agent then independently solves its own receding-horizon control problem conditioned on the assignment.

Beyond the choice of game formulation, the pursuer team's desired behavior is itself mission-dependent. Capturing and containing an evader requires different coordination than preventing it from reaching a protected region. Counter-UAV defense provides a motivating example: capture may be preferred for forensic analysis and attribution, whereas an imminent threat may require immediate neutralization. We formalize this flexibility through mission-dependent rules-of-engagement (RoE), which determine each pursuer's tactical posture (DEFEND, NEUTRALIZE, CAPTURE, RECON). The strategic planner can therefore adapt how team resources are allocated as the tactical situation evolves, without changing the underlying control architecture.

In short, to enable different engagement modes without redesigning the underlying control logic, we propose an engagement-aware hierarchical control architecture that decouples strategic role assignment from low-level safety and engagement enforcement (see Fig. 1): the LLM provides the strategic role assignment, each agent's model predictive control (MPC) translates that role into a nominal control action, and a control barrier function (CBF)-based filter enforces safety and role-dependent engagement constraints Critically, the assigned role parameterizes both the MPC cost and the CBF constraint set. For example, assigning NEUTRALIZE removes the pursuer –evader standoff constraint, permitting direct contact. Thus, switching from stand-off capture to direct neutralization requires only a change in the assigned role, which reparameterizes the MPC objective and active CBF constraints while preserving the same underlying control architecture. Our contributions are as follows:

- An engagement-aware pursuit-evasion framework that treats rules of engagement as online tactical decision variables rather than fixed a priori;
- An LLM-based strategic planning layer that assigns interpretable, adaptive tactical roles to each agent under field-of-view-limited observations, with an explicit natural-language justification;
- A hierarchical execution architecture in which per-agent MPC controllers and CBF-based safety filters realize LLM-assigned roles without redesigning the underlying control architecture;
- A simulation-based benchmark for zero-shot LLM strategic planning in partially observed, adversarial multi-agent settings, including state-based tactical reasoning and image-based contact classification.

2 Related Work

Large language models have increasingly been adopted as high-level planners for embodied agents, translating natural-language task descriptions and environment state into executable action sequences without task-specific training. Early work grounded a language model's proposed actions in a robot's actual affordances (Ahn et al. 2022), used a language model directly as a zero-shot sequential planner over admissible actions (Huang et al. 2022), and generated executable robot code or structured task programs from natural-language instructions (Liang et al. 2023; Singh et al. 2022). More recent work extends this paradigm to multi-robot settings, where a language model coordinates several embodied agents through natural-language dialogue (Mandi, Jain, and Song 2024) or is compared directly against a decentralized per-robot instantiation to characterize the centralization-communication tradeoff in LLM-driven multi-robot coordination (Chen et al. 2024).

Multi-agent pursuit-evasion has a broad literature spanning differential games, optimization-based control, reinforcement learning, and distributed multi-robot coordination. Prior work has studied heterogeneous pursuer–evader teams under uncertainty (Zhang, Prorok, and Bhattacharya 2021), cooperative multi-UAV target defense (Tong and Duan 2023), reinforcement learning for decentralized pursuit (Zhang et al. 2022), distributional soft actor-critic methods for underwater vehicle pursuit (Hou et al. 2023), limited-information model predictive control for pursuit-evasion games (Sani, Robu, and Hably 2021), evader-agnostic pursuit in partially observable environments (Kalanther et al. 2025), and online deep reinforcement learning for multi-UAV pursuit in unknown 3D environments (Chen et al. 2025). Distributed optimization frameworks provide another important direction, enabling multi-robot systems to make real-time local decisions that collectively achieve global objectives such as area coverage and target assignment (Jaleel and Shamma 2020).

Several works explicitly address prediction, uncertainty, and geometric guarantees. Shivam et al. (Shivam et al. 2019) propose a predictive tracking framework based on Newton–Raphson flow, combined with an online-trained neural network to estimate evader strategies. GRAPE (Shah and Schwager 2019) models noisy evader localization using uncertainty ellipsoids and minimizes the evader's safe-reachable set, while Wang et al. (Wang et al. 2025) formulate encirclement under unknown evader motion using distributed robust MPC.

Model-based and model-free distributed optimization methods provide another foundation for cooperative pursuit. Zhou et al. (Zhou and Chen 2023) formulate pursuit as an infinite-horizon differential game over graph-based formation and tracking errors, yielding Nash-equilibrium policies through Hamilton–Jacobi equations but requiring full observability and known dynamics. Dong et al. (Dong, Zhang, and Ming 2024) reduce this dependence on explicit dynamics using a Q-learning and actor-critic formulation for linear-quadratic systems, though such approaches typically require sufficient exploration and introduce additional training complexity. Safety-critical pursuit has also been studied through visibility and barrier-function constraints. Zhou et al. (Zhou et al. 2025) maintain a moving evader within a pursuer's field of view under occlusions by combining a CBF constraint with sampling-based planning.

This work separates strategic planning from low-level control: two independent LLM planners assign team-specific roles and targets under partial observability, while each robot independently executes its assignment through role-conditioned MPC and CBF-based safety filtering. We evaluate this hierarchy using both state-based tactical reasoning and image-based contact classification.

3 Approach

Our architecture decomposes the multi-agent interaction into three hierarchical layers: a strategic planner, realized as an LLM agent, that assigns each agent a rule of engagement; a decentralized receding-horizon MPC trajectory generator that independently realizes every agent's currently assigned role; and a discrete-time CBF-QP safety and engagement filter that enforces inter-agent and adversarial constraints on the resulting commands.

3.1 Rules of Engagement

We consider pursuit-evasion scenarios where the pursuer team operates under mission-dependent RoE rather than pursuing a static objective. While conventional formulations treat capture as the sole terminal goal, direct capture can be infeasible or disadvantageous when the evader has superior mobility or operates near sensitive regions where aggressive interception introduces collateral risk. In these settings, the pursuer team must dynamically transition across alternative engagement modes, including perimeter defense and neutralization.

Counter-UAV missions provide concrete motivation for this operational flexibility. When safe recovery is viable, capture is often preferred because it facilitates forensic analysis, attribution, and technical exploitation of the target's avionics, communication links, navigation payload, and hardware. Analysis of systems such as Shahed/Geran drones and the development of platforms like LUCAS underscore the intelligence and operational value of recovering adversarial UAVs (Center for Strategic and International Studies, Missile Defense Project 2026). Conversely, when recovery is impractical or the target presents an imminent threat, the operational priority shifts to neutralization. This approach is exemplified in Ukraine by counter-UAV platforms such as the P1-SUN, Sting, and ZIRKA, which are designed to intercept hostile UAVs.

We implement RoE adaptation through discrete, per-agent roles assigned at each planning cycle by a high-level strategic planner – realized in this work via an LLM. Each role (DEFEND, NEUTRALIZE, CAPTURE, RECON) jointly dictates the receding-horizon MPC objective and, where relevant, the low-level CBF safety constraints, realizing these operational modes end-to-end without modifying the underlying control architecture. Under the *defend* role, agents maintain designated stations along the protected perimeter, enforcing the deterrence posture. The *recon* role instead diverts an agent to actively patrol toward a coverage region and sweep its sensor heading independently of its direction of travel, at no change to the CBF constraints – it trades perimeter presence or engagement capacity for expanded situational awareness. Both *neutralize* and *capture* instead drive agents toward the target using an identical interception MPC cost, differing solely in their CBF safety filtering. An agent assigned to *neutralize* is exempted from the pursuer-evader separation constraint $D_{\text{safe, pe}}$, permitting direct kinetic contact that immediately terminates the target at the cost of permanently expending the intercepting agent. In *capture* mode, $D_{\text{safe, pe}}$ remains strictly enforced; the target is captured via multi-agent containment, requiring at least κ capturing agents to simultaneously enter the target's capture radius (we set $\kappa = 2$ in our experiments [cf. (Bopardikar and Suri 2014)]) – a coordinated maneuver that preserves all pursuer assets at the expense of time and agent allocation. This deliberate asymmetry forces the high-level strategic planner to weigh rapid target elimination against asset

preservation, evaluated dynamically via the scoring formulation in Appendix A based on asset availability and threat urgency.

The simulation environment used here is intentionally low-fidelity: agents move under simplified double-integrator translational dynamics in a 2D arena, sensing is a noiseless conic field-of-view with perfect detection, and no communication delay or environmental disturbance is modeled. We make no claim that this setup is representative of real-world multi-robot deployment. The point of this work is narrower: to show that off-the-shelf large language models, used entirely zero-shot – with no fine-tuning, task-specific training data, or reinforcement learning – can serve as effective strategic planners for multi-agent tactical decision-making, producing interpretable, justified role assignments under a partially observed, adversarial state. Demonstrating this capability in a simplified but genuinely multi-agent, partially-observable, adversarial setting is a necessary first step before tackling the added complexity of higher-fidelity dynamics, sensing, or real hardware, which we leave to future work.

The interface between the strategic and execution layers is deliberately narrow. The strategic layer communicates with the execution layer through a single tuple per agent per cycle, a role and an optional target id, from which each agent's MPC cost and CBF constraint set are instantiated, accompanied by a one-sentence natural-language justification. This orchestrator-worker pattern (a planner assigning discrete role/target tuples to independently-executing workers) is not specific to pursuit-evasion; it mirrors the broader agentic-AI pattern of a coordinator delegating typed subtasks to specialized executors.

3.2 Agent Dynamics and State Space

Every agent, pursuer or evader, evolves under the following discrete-time control-affine dynamics, discretized at a fixed timestep dt via semi-implicit Euler integration:

$$x_{k+1} = f(x_k) + g\,u_k,$$

where the state x and control input u are

$$x = \begin{bmatrix} p^\top & v^\top & \theta \end{bmatrix}^\top \in \mathbb{R}^5, \qquad u = \begin{bmatrix} a^\top & \omega \end{bmatrix}^\top \in \mathbb{R}^3,$$

with $p, v \in \mathbb{R}^2$ its planar position and velocity, $\theta \in \mathbb{R}$ its heading, $a \in \mathbb{R}^2$ a commanded translational acceleration, and $\omega \in \mathbb{R}$ an independent commanded yaw rate. The drift and input matrices are

$$f(x_k) = \begin{bmatrix} p_k + dt\, v_k \\ v_k \\ \theta_k \end{bmatrix}, \qquad g = \begin{bmatrix} dt^2 I_2 & \mathbf{0}_{2\times 1} \\ dt\, I_2 & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & dt \end{bmatrix} \in \mathbb{R}^{5\times 3},$$

where I_2 is the 2×2 identity matrix. Heading evolves independently of position and velocity, such as a sensor turret mounted on a vehicle (e.g., rotating 360 camera). All controls and velocities are box-constrained per axis – $|a_x|, |a_y| \le a_{\max}$, $|v_x|, |v_y| \le v_{\max}$, and $|\omega| \le \omega_{\max}$. $p_{i,k}, v_{i,k} \in \mathbb{R}^2$ denote agent i's current position and velocity at time step k. Moreover, $p_{ij} = p_i - p_j$, $v_{ij} = v_i - v_j$ denote the relative position and velocity between agent i and j.

3.3 Decentralized Receding-Horizon MPC

We define two disjoint families of receding horizon control problems for each team: the pursuer (defending) team's four roles – DEFEND, RECON, NEUTRALIZE, CAPTURE – and the evader (attacking) team's two roles – ATTACK, EVADE. All six share the same receding-horizon template below, differing only in the role-specific cost term J_{role}; we define that template first, then J_{role} for each pursuer role, then for each evader role.

Every pursuer i (assigned role DEFEND, RECON, NEUTRALIZE, or CAPTURE) and every evader j (assigned role ATTACK or EVADE) solves, from its current state x_0,

$$U^* = \arg\min_U \; w_u \sum_{k=0}^{N-1} \|u_k\|^2 + J_{\text{role}}(X)$$

subject to its own dynamics 3.2, x_0 fixed to its current measured state, and actuation/velocity bounds $\|u_k\|_\infty \le a_{\max}$, $\|v_k\|_\infty \le v_{\max}$, enforced independently per axis, where $w_{\text{u}} > 0$ is a fixed weight penalizing control effort. $U = (u_0, \ldots, u_{N-1})$, $X = (x_1, \ldots, x_N)$ are the resulting control and state sequences over the horizon. Only the role-specific term J_{role} changes across roles. For roles whose cost is defined relative to the opposing team (DEFEND, NEUTRALIZE, CAPTURE, EVADE), this also determines which specific opposing agent it is solved against; RECON and ATTACK instead target a patrol waypoint, with no dependence on the opposing team's state at all.

Defend A pursuer assigned DEFEND holds a guard station on the perimeter of the protected region rather than closing on the evader. At each planning step it targets the closest point p_{bound} on the (polygon-approximated) defended boundary to the nearest evader's *current* position, and minimizes

$$J_{\text{defend}} = \sum_{k=1}^{N} w_{\text{defend}} \|p_{i,k} - p_{\text{bound}}\|^2.$$

Recon A pursuer assigned RECON is diverted from perimeter duty or engagement entirely to actively scout the arena, with no dependence on the evader's state. Its patrol target p_{target} is a coverage-region center supplied by the strategic layer. Coverage regions are defined by discretizing the arena into a coarse fixed grid; each cell's staleness is the number of planning cycles elapsed since any pursuer's field of view last covered it, and the strategic layer assigns the region with the highest staleness (never-covered cells ranked above any finite value) as the target. Once reached, a directed target is held, while a random one is redrawn. The pursuer minimizes a hold-position term identical in form to DEFEND's,

$$J_{\text{hold}} = \sum_{k=1}^{N} w_{\text{hold}} \|p_{i,k} - p_{\text{target}}\|^2,$$

together with an independent bearing-sweep term that rotates its sensor heading $\theta_{i,k}$ sinusoidally about the direction of travel $\theta_c = \angle(p_{\text{target}} - p_{i,0})$, over a $2\pi/3$ arc and a fixed

sweep period T_s:

$$J_{\text{sweep}} = \sum_{k=1}^{N} w_{\text{sweep}} \left(\theta_{i,k} - \theta_c - \tfrac{\pi}{3}\sin\!\left(\tfrac{2\pi(t_0+k\,dt)}{T_s}\right)\right)^2,$$

so that $J_{\text{recon}} = J_{\text{hold}} + J_{\text{sweep}}$. Translation and heading are governed by independent controls (Section 3.2), so the sweep is a genuine rotation of the sensor's field of view while the pursuer is in transit, not merely a byproduct of its direction of motion.

Neutralize and Capture NEUTRALIZE and CAPTURE pursuers run the same intercept controller against their nearest evader; the roles differ only downstream, in whether the CBF enforces or exempts the pursuer-evader standoff constraint for that agent. Since the evader is not solved jointly, its future trajectory is predicted open-loop under a constant-velocity assumption, $\bar{p}_{e,k} = p_{e,0} + k\,dt\,v_{e,0}$, and the pursuer minimizes

$$J_{\text{intercept}} = \sum_{k=1}^{N} w_e \|p_{i,k} - \bar{p}_{e,k}\|^2,$$

tracking this predicted position directly rather than also matching a velocity-cutoff term against it; recomputing $\bar{p}_{e,k}$ from the evader's current state at every planning step still lets the controller correct for prediction drift each cycle, without committing to a closing velocity derived from the same stale linear extrapolation. An evader is captured once at least κ CAPTURE-role pursuers are simultaneously within the capture radius r_c of it, or immediately upon direct contact with a single NEUTRALIZE-role pursuer.

Attack and Evade An evader assigned ATTACK advances directly on the defended region's center p_{def}, independent of the pursuers' plans:

$$J_{\text{attack}} = \sum_{k=1}^{N} w_{\text{prog}} \|p_{e,k} - p_{\text{def}}\|^2.$$

An evader assigned EVADE instead reacts to the pursuers' just-computed nominal plans $\{\bar{X}_{p,i}\}_{i=1}^{N_p}$, maximizing its predicted separation from every pursuer's planned trajectory over the horizon:

$$J_{\text{evade}} = -w_{\text{avoid}} \sum_{k=1}^{N} \sum_{i=1}^{N_p} \|p_{e,k} - \bar{p}_{i,k}\|^2.$$

Because the avoidance term is a concave quadratic, this problem is solved with a general nonlinear solver, unlike the convex QPs used for the other three roles.

3.4 Discrete-Time CBF Safety Filter

Each agent's role-conditioned MPC acts as a mid-level, task-oriented planner, while CBFs (Ames et al. 2019; Dhiman et al. 2023) serve as a low-level, one-step safety filter applied downstream of the MPC, independently of which role generated the nominal command. Under mild conditions, a CBF-based filter is necessary and sufficient to render a safe set forward-invariant, guaranteeing constraint satisfaction regardless of the nominal controller that generated the desired input (Ames et al. 2019); following this established practice, our discrete-time CBF filter minimally invasively corrects each agent's nominal MPC command to preserve safety. To ensure collision avoidance between two agents i and j, we define a velocity-aware kinematic barrier function $h(x_k)$ (Cheng et al. 2020; Ataei, Dhiman, and Khojasteh 2025; Borrmann et al. 2015),

$$h(x_k) = \frac{p_{ij}^T v_{ij}}{\|p_{ij}\|} + \sqrt{\alpha_{\max}(\|p_{ij}\| - D_{\text{safe}})} \geq 0$$

where p_{ij} and v_{ij} are defined in Section 3.2. Here D_{safe} is the minimum separation required between agents i and j – the pair-specific values used in this paper ($D_{\text{safe,p}}$ for pursuer-pursuer pairs, $D_{\text{safe,pe}}$ for pursuer-evader pairs, $D_{\text{safe,e}}$ for the evader's own filter). Under appropriate assumptions on $\alpha_{\max}$ stated in Lemma 2 of Cheng et al. (2020), the CBF defined above renders the safe set invariant.

Given the desired optimal control input u_{des} generated by the agent's MPC controller, the true control input applied to the system is the solution to a constrained Quadratic Program (QP) with decision variables u and slack variables ϵ:

$$u^*, \epsilon^* = \arg\min_{u,\epsilon} \frac{1}{2}\|u - u_{\text{des}}\|^2 + \lambda\epsilon^2$$

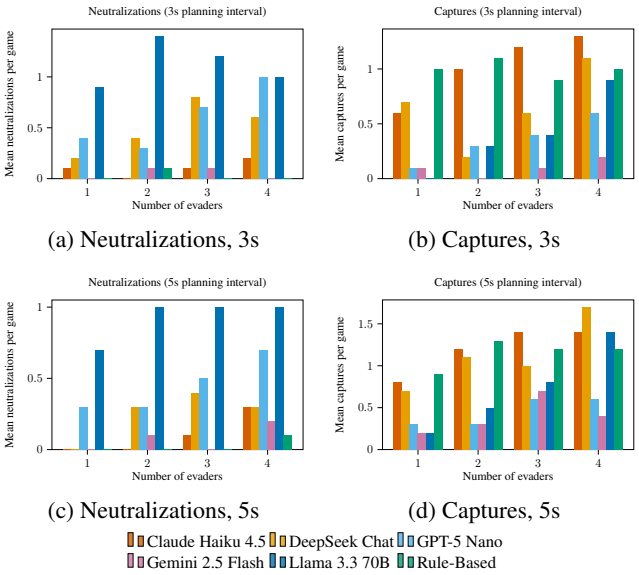
Because ϵ is a free slack variable rather than hard-constrained to zero, the CBF condition above is enforced as a penalized soft constraint. We implement these constraints for *pursuer-pursuer* collision avoidance ($D_{\text{safe, p}}$) and *pursuer-evader* standoff ($D_{\text{safe, pe}}$); the latter is omitted entirely for a pursuer assigned NEUTRALIZE, permitting the direct contact that role requires. A mirrored constraint ($D_{\text{safe, e}}$) is enforced on the evader's own control update.

4 Experiments

To evaluate the performance of the proposed architecture, we conducted simulations within a 2D planar double integrator dynamics framework. Each agent's receding-horizon optimization problem is modeled using CasADi and solved independently: OSQP for the convex quadratic formulations used by the DEFEND, RECON, NEUTRALIZE/CAPTURE, and ATTACK controllers, and IPOPT for the nonconvex formulation used by EVADE. The lower level discrete CBF safety filter bypasses this CasADi modeling layer entirely, assembling its sparse QP directly and solving it via OSQP's native interface, to maintain high control frequencies.

All experiments feature a team of four pursuers under decentralized control and central planning, defending against an evader presence whose scale we vary across trials – from a single evader up to four in the state-only setting, or from one to four contact slots in the multi-modal setting (Section 4.2), where not every slot is guaranteed to resolve to a genuine evader.

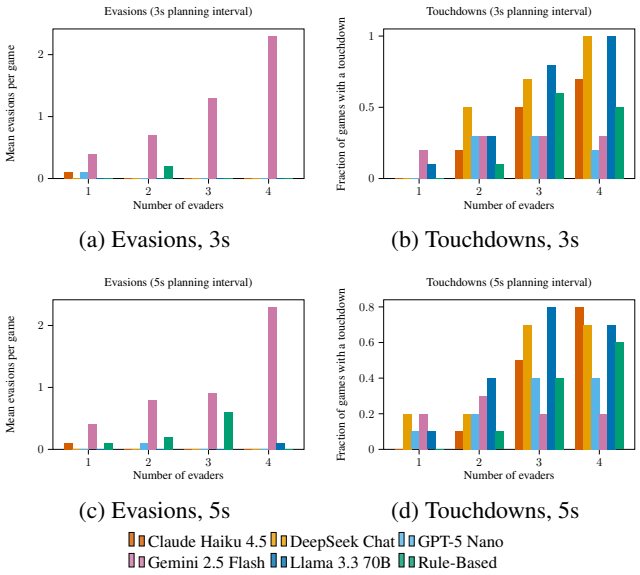
Two layers of fault tolerance keep a single unreliable model from invalidating a run: within a cycle, a role assignment that violates the structured-output schema is retried, and on repeated failure the affected team falls back to a fixed



(a) Neutralizations, 3s (b) Captures, 3s

(c) Neutralizations, 5s (d) Captures, 5s

Figure 2: Experiment I: mean number of pursuer-initiated eliminations per game as a function of the number of evaders, by model, split by planning interval (top row: 3s; bottom row: 5s). NEUTRALIZE (a, c) resolves fast with a single agent at the cost of permanently disabling it, while CAPTURE (b, d) requires two or more agents to converge but preserves the whole force; the two roles are directly comparable at every evader count since both share the same underlying scoring, and the two rows isolate how much of that behavior is sensitive to how often the RoE layer gets to replan.



(a) Evasions, 3s (b) Touchdowns, 3s

(c) Evasions, 5s (d) Touchdowns, 5s

Figure 3: Experiment I: defensive failure modes as a function of the number of evaders, by model, split by planning interval (top row: 3s; bottom row: 5s). Evasions (a, c) count evaders that leave the arena unresolved, reported as a mean per game; touchdowns (b, d) report the fraction of games in which the defended region was breached, an episode-ending outcome capped at one per game.

safe default (DEFEND for blue, ATTACK for red) for that cycle only, logged as a fallback event; at the episode level, an unhandled LLM API failure is recorded as that episode's outcome.

Within this dynamic framework, we evaluate the strategic planner under two observational modalities. In the state-information-only modality, the LLM planning agent observes only agent positions, velocities, and status. In the multi-modal modality, the agent additionally receives a static reference image of each detected contact – a deliberate abstraction of perception rather than a physically-grounded rendering of the simulated scene, since the image carries no dependence on a contact's actual range, viewing angle, or motion, nor any sensor noise; Section 4.2 discusses this abstraction and its implications for interpreting the resulting discrimination performance in full.

4.1 State-Only Strategic Planning

Before introducing visual input, we first establish how the agentic RoE Planner performs when it reasons purely over structured state – positions, velocities, and role/status flags for every agent. We sweep three axes against this state-only planner. The blue team is fixed at four pursuers; we vary (i) the size of the red team, $N_e \in 1, 2, 3, 4$, (ii) the planner's replanning latency, $3, 5$ simulated seconds, and (iii) the model instantiating both the blue and red RoE planners. All other simulation parameters are held fixed across the sweep: a $dt = 0.1$, s integration step, an $N = 20$-step MPC horizon, a 600-step $(60, \text{s})$ episode budget, and both the pursuer-pursuer and pursuer-evader CBF filters enabled. For each of the 4 red-team sizes we run 10 independently seeded trials, randomizing every agent's initial position and velocity (blues around the defended perimeter, reds near the corner of the arena diametrically opposite it); this is repeated for every combination of the 2 planning latencies and 6 candidate models – DeepSeek V3, Qwen3-30B, Claude Haiku 4.5, GPT-5 Nano, Llama 3.3 70B, and Gemini 2.5 Flash.

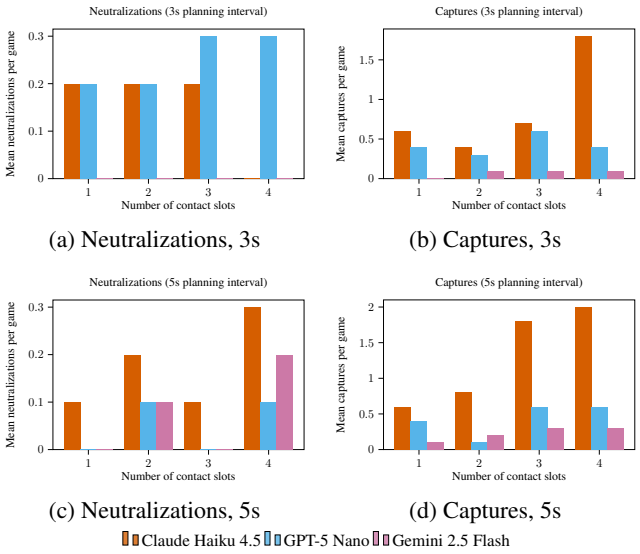
Each episode terminates on one of: a perimeter breach, a blue-blue collision, a solver failure at either the per-agent MPC or CBF-QP stage, or resolution of every red agent via capture, neutralization, or escape; an episode that exhausts its step budget without any of these is logged as incomplete. For every episode we record the terminal outcome, steps completed, number of reds captured versus escaped, the resulting blue score (Appendix A.1), and the number of blues permanently disabled via NEUTRALIZE. Independently of outcome, every planning cycle in every episode is logged at the level of individual agents: for each active blue and red, the role it was assigned and the model's one-sentence natural-language justification for that assignment, giving a complete, per-decision reasoning trace across the full sweep rather than only aggregate statistics.

Histograms of all four end conditions per agent are shown in Figure 2 and Figure 3. Qwen3-30B-A3B is excluded from

these results because it failed to reliably produce schema-compliant structured output, forcing the safe-default fallback on a large fraction of planning cycles for both teams. The rule-based heuristic planner for the attacking team assigns EVADE once a detected blue agent is within a fixed minimum distance of that specific red, and ATTACK otherwise A.1. This baseline is competitive with several LLMs: it achieves the second-highest capture rate of any model (behind only ClaudeHaiku4.5), a touchdown rate lower than three of the five LLMs, and few neutralizations, indicating that its threats are resolved overwhelmingly through multi-agent CAPTURE rather than single-agent contact. Among the five LLMs, tactical style diverges sharply despite the identical role vocabulary: Llama3.370B relies overwhelmingly on NEUTRALIZE, trading a permanently disabled agent for a fast single-agent kill, yet still posts the highest touchdown rate of any model, suggesting this aggression comes at the expense of perimeter coverage. ClaudeHaiku4.5 and DeepSeekChat instead favor CAPTURE, preserving their force at the cost of committing two agents per engagement – though the two diverge sharply in outcome, with Claude's touchdown rate middling and DeepSeek's nearly as high as Llama's, so favoring CAPTURE over NEUTRALIZE does not by itself predict defensive success. GPT-5Nano achieves the lowest touchdown rate overall despite only moderate engagement volume, indicating its assignments are more efficiently timed or targeted rather than simply more numerous. Gemini2.5Flash shows the most evasions among the LLMs – more reds are left to exit the arena unaddressed – yet still holds a comparably low touchdown rate, consistent with a containment-oriented strategy that tolerates some threats escaping rather than overcommitting the defense.

4.2 Multi-Modal Strategic Planning

The task this perception is meant to support is genuine visual discrimination, not a trivial one. Since this requires interpreting an attached image, we restrict this experiment to the three of the six aforementioned models capable of multimodal (image) input – Claude Haiku 4.5, GPT-5 Nano, and Gemini 2.5 Flash; the remaining three, which accept text only, are not applicable here. Rather than a fixed red team, each of N_{slot} contact slots independently resolves, once per episode, to one of three kinds: a real antagonistic drone (probability 0.7, a genuine threat and the only kind counted toward the win condition), a harmless bird (probability $p_b = 0.15$), or a harmless civilian drone (probability $p_c = 0.15$). Reds and decoy contacts are shuffled into a single unlabeled contacts roster, so the model never sees which slot is which – it must judge each contact's attached image on its own. Unlike the bird, which is silhouette-distinguishable at a glance, the civilian drone is deliberately the hard case: it is a quadcopter too, and can look very similar to the antagonistic drone, so the planner must rely on cues an image genuinely carries (e.g. color, markings, build) rather than shape alone. To guarantee every episode has a resolvable objective, the per-slot draw is resampled until at least one slot resolves to a real drone; the realized composition (how many of each kind) is logged per episode rather than assumed from N_{slot} alone, since it is a random outcome of this draw and not a fixed count.



(a) Neutralizations, 3s (b) Captures, 3s

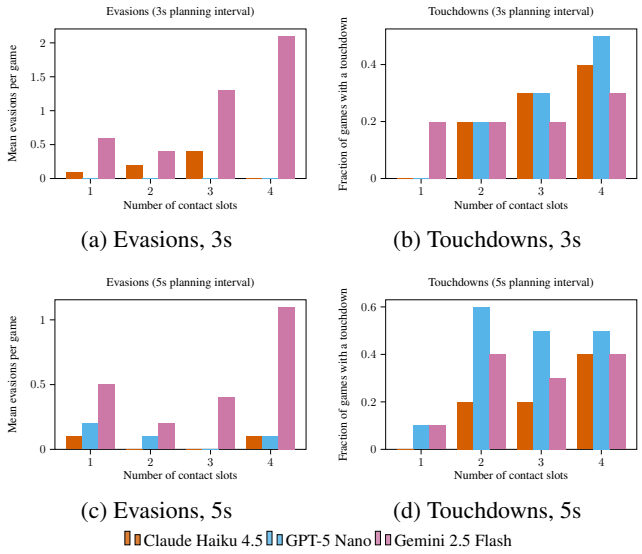
(c) Neutralizations, 5s (d) Captures, 5s

Figure 4: Experiment II: mean number of pursuer-initiated eliminations per game as a function of the number of contact slots offered, by vision-capable model, split by planning interval (top row: 3s; bottom row: 5s).

Histograms of all four end conditions per agent, aggregated over contact-slot counts, are shown in Figure 4 and Figure 5. Unlike experiment I, no model required exclusion here – schema-compliance failures were negligible across all three vision-capable models. ClaudeHaiku4.5 and GPT-5Nano account for the large majority of neutralizations and captures between them, with Claude in particular achieving both the highest capture rate of any model and the lowest touchdown rate, indicating an effective, well-targeted engagement policy despite the added visual-discrimination burden. Gemini2.5Flash, by contrast, both engages far less often and evades far more – 0.825 mean evasions per game against at most 0.113 for the other two – and the reasoning logs help explain why: across its 80 games, Gemini commits an agent to a civilian drone 134 times, more than either other model, despite completing the fewest real captures and neutralizations overall, so a disproportionate share of its already-sparse engagement attempts are misdirected at the harder decoy category rather than the genuine threat. This asymmetry holds across all three models: bird engagements are essentially absent (at most one, across 240 games total), while civilian engagements number in the dozens to low hundreds per model, confirming that the bird decoy is reliably ruled out by its winged silhouette as intended, while the visually similar civilian drone remains a genuine source of confusion even for the strongest models.

5 Conclusion and Future Work

This work demonstrates a control architecture for multi-agent adversarial pursuit-evasion that combines decentralized control with centralized, per-team strategic planning. Each agent's motion is governed by its own independent



(a) Evasions, 3s (b) Touchdowns, 3s

(c) Evasions, 5s (d) Touchdowns, 5s

Figure 5: Experiment II: defensive failure modes as a function of the number of contact slots offered, by vision-capable model, split by planning interval (top row: 3s; bottom row: 5s).

model-predictive controller, filtered through a control barrier function safety layer, while a single centralized planner per team assigns every active agent on that team a role, re-evaluated on a slower cadence than the underlying decentralized control loop. We instantiate this centralized planning layer a LLM operating over structured, natural-language tactical observations, and evaluate it where each team's planner only perceives opposing agents within its combined field of view, rather than assuming full state knowledge. We further extend this to a multi-modal setting in which the planner is given an image of each detected contact and must visually distinguish genuine threats from harmless decoys before committing to an engagement. Across both the state-only and vision-augmented settings, the framework demonstrates that an LLM can serve as an effective centralized tactical orchestrator. The underlying simulation is intentionally low-fidelity: the framework is best understood as a benchmark for LLM-driven agentic planning in cyber-physical systems, rather than a deployable pursuit-evasion controller in itself. Moving to a higher-fidelity environment is left to future work.

"Contact 1 appears to be a harmless civilian drone, and with two agents committed to a high-priority threat, agent 2 should hold a defensive position near the perimeter."

(a) Correct identification

"Agent 0 will CAPTURE contact 3 because it appears to be a hostile drone, and capturing allows us to preserve our asset while still neutralizing the threat."

(b) Failed identification

Figure 6: Contrasting outcomes on the harder decoy case (`google/gemini-2.5-flash`, 3 s planning interval). Both contacts are confirmed civilian drones by ground truth. In (a), the planner correctly reads the image as a harmless civilian drone and assigns DEFEND instead of engaging. In (b), the planner misreads the visually similar civilian drone as a hostile target and commits an agent to CAPTURE against it – a wasted engagement, since two agents are needed to converge on a target that poses no actual threat.

References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Ames, A. D.; Coogan, S.; Egerstedt, M.; Notomista, G.; Sreenath, K.; and Tabuada, P. 2019. Control Barrier Functions: Theory and Applications. In *2019 18th European Control Conference (ECC)*, 3420–3431.

Ataei, M.; Dhiman, V.; and Khojasteh, M. J. 2025. K-DAREK: Distance Aware Error for Kurkova Kolmogorov Networks. In *2025 59th Asilomar Conference on Signals, Systems, and Computers*, 1336–1342.

Bopardikar, S. D.; and Suri, S. 2014. k-Capture in multiagent pursuit evasion, or the lion and the hyenas. *Theoretical Computer Science*, 522: 13–23.

Borrmann, U.; Wang, L.; Ames, A. D.; and Egerstedt, M. 2015. Control Barrier Certificates for Safe Swarm Behavior. *IFAC Conference on Analysis and Design of Hybrid Systems*.

Center for Strategic and International Studies, Missile Defense Project. 2026. Shahed-131 and -136. *Missile Threat*. Last updated May 22, 2026. [Online]. Available: https://missilethreat.csis.org/missile/shahed-131-and-136/.

Chen, J.; Yu, C.; Li, G.; Tang, W.; Ji, S.; Yang, X.; Xu, B.; Yang, H.; and Wang, Y. 2025. Online planning for multi-uav pursuit-evasion in unknown environments using deep reinforcement learning. *IEEE Robotics and Automation Letters*.

Chen, Y.; Arkin, J.; Zhang, Y.; Roy, N.; and Fan, C. 2024. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 4311–4317. IEEE.

Cheng, R.; Khojasteh, M. J.; Ames, A. D.; and Burdick, J. W. 2020. Safe multi-agent interaction through robust control barrier functions with learned uncertainties. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 777–783. IEEE.

Chung, T. H.; Hollinger, G. A.; and Isler, V. 2011. Search and pursuit-evasion in mobile robotics: A survey. *Autonomous robots*, 31(4): 299–316.

Dhiman, V.; Khojasteh, M. J.; Franceschetti, M.; and Atanasov, N. 2023. Control Barriers in Bayesian Learning of System Dynamics. *IEEE Transactions on Automatic Control*, 68(1): 214–229.

Dong, X.; Zhang, H.; and Ming, Z. 2024. Adaptive optimal control via Q-learning for multi-agent pursuit-evasion games. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 71(6): 3056–3060.

Firoozi, R.; Tucker, J.; Tian, S.; Majumdar, A.; Sun, J.; Liu, W.; Zhu, Y.; Song, S.; Kapoor, A.; Hausman, K.; et al. 2025. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5): 701–739.

Hou, Y.; Han, G.; Zhang, F.; Lin, C.; Peng, J.; and Liu, L. 2023. Distributional soft actor-critic-based multi-AUV cooperative pursuit for maritime security protection. *IEEE Transactions on Intelligent Transportation Systems*, 25(6): 6049–6060.

Hu, Y.; Xie, Q.; Jain, V.; Francis, J.; Patrikar, J.; Keetha, N.; Kim, S.; Xie, Y.; Zhang, T.; Fang, H.-S.; et al. 2023. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, 9118–9147. PMLR.

Jaleel, H.; and Shamma, J. S. 2020. Distributed optimization for robot networks: From real-time convex optimization to game-theoretic self-organization. *Proceedings of the IEEE*, 108(11): 1953–1967.

Jülg, T.; Burgard, W.; and Walter, F. 2025. Refined policy distillation: From vla generalists to rl experts. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 11677–11684. IEEE.

Kalanther, A.; Bostwick, D.; Maheshwari, C.; and Sastry, S. 2025. Evader-Agnostic Team-Based Pursuit Strategies in Partially-Observable Environments. *arXiv preprint arXiv:2511.05812*.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, 9493–9500. IEEE.

Mandi, Z.; Jain, S.; and Song, S. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 286–299. IEEE.

Matni, N.; Ames, A. D.; and Doyle, J. C. 2024. Towards a theory of control architecture: A quantitative framework for layered multi-rate control. *arXiv preprint arXiv:2401.15185*.

Navaratnarajah, S.; Kim, T.; Ruthardt, J.; Bhimwal, I.; Yamada, R.; Blei, Y.; Burgard, W.; and Asano, Y. M. 2026. Zero-Shot Mission-Level Evaluation for Aerial MLLM Agents. *arXiv preprint arXiv:2607.22014*.

Sani, M.; Robu, B.; and Hably, A. 2021. Limited information model predictive control for pursuit-evasion games. In *2021 60th IEEE Conference on Decision and Control (CDC)*, 265–270. IEEE.

Shah, K.; and Schwager, M. 2019. Grape: Geometric risk-aware pursuit-evasion. *Robotics and Autonomous Systems*, 121: 103246.

Shishika, D.; and Kumar, V. 2020. A review of multi agent perimeter defense games. In *International conference on decision and game theory for security*, 472–485. Springer.

Shivam, S.; Kanellopoulos, A.; Vamvoudakis, K. G.; and Wardi, Y. 2019. A predictive deep learning approach to output regulation: The case of collaborative pursuit evasion. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 853–859. IEEE.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2022. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*.

Tong, B.; and Duan, H. 2023. A game theory-based approach for multiple UAVs cooperative target defense. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 71(4): 2149–2153.

Wang, C.; Chen, H.; Pan, J.; and Zhang, W. 2025. Distributed cooperative pursuit with encirclement guarantee via robust model predictive control. *Robotics and Autonomous Systems*, 192: 105019.

Weintraub, I. E.; Pachter, M.; and Garcia, E. 2020. An introduction to pursuit-evasion differential games. In *2020 American Control Conference (ACC)*, 1049–1066. IEEE.

Weintraub, I. E.; Von Moll, A.; and Pachter, M. 2023. Range-limited pursuit-evasion. In *NAECON 2023-IEEE National Aerospace and Electronics Conference*, 28–35. IEEE.

Zhang, L.; Prorok, A.; and Bhattacharya, S. 2021. Pursuer assignment and control strategies in multi-agent pursuit-evasion under uncertainties. *Frontiers in Robotics and AI*, 8: 691637.

Zhang, Z.; Wang, X.; Zhang, Q.; and Hu, T. 2022. Multi-robot cooperative pursuit via potential field-enhanced reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, 8808–8814. IEEE.

Zhou, M.; Shaikh, M.; Chaubey, V.; Haggerty, P.; Koga, S.; Panagou, D.; and Atanasov, N. 2025. Control strategies for pursuit-evasion under occlusion using visibility and safety Barrier functions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 12863–12869. IEEE.

Zhou, P.; and Chen, B. M. 2023. Distributed optimal solutions for multiagent pursuit-evasion games for capture and formation control. *IEEE Transactions on Industrial Electronics*, 71(5): 5224–5234.

A System Prompts

Both teams' strategic planning layers are realized as an LLM called once per planning cycle with a fixed system prompt (below) and a per-cycle user message containing the current tactical observation (Section 4). The model's response is constrained to a strict JSON schema (Table 1); the prompt text and the schema jointly determine what the model can express, so both are reproduced here for completeness. Placeholder values in the BLUE prompt (`capture_points_neutralize`, `capture_points_capture`, `capture_threshold`) are filled in at construction time from the same constants that govern the simulation's actual scoring, so the model is never told an incentive structure that diverges from what it is actually evaluated against; the values shown below are the defaults used in all reported experiments.

A.1 Experiment I

Defending Team (BLUE)

```
You are the tactical commander for the
BLUE team defending a region against
RED intruders in a 2D pursuit-evasion
simulation. Each planning cycle, you
assign every active blue agent exactly
one role:
- DEFEND: hold a guard station on the
perimeter of the defended region.

- NEUTRALIZE: a single blue closes
to physical contact with a red agent.
No collision-avoidance safety margin
is applied against red for this role
- contact is allowed and intended.
Resolves fast, with just one agent.
Consequence: that blue is permanently
disabled afterward (it takes no further
part in the engagement) and the kill is
worth only 1.0 points.

- CAPTURE: 2 or more blues converge
on the same red agent simultaneously
and hold within capture_radius while
maintaining a collision-avoidance
safety margin - red is captured
by proximity without any contact.
Slower and costlier up front (2
committed agents instead of 1), but
every participating blue remains
fully operational afterward, and the
kill is worth 2.0 points - more than
NEUTRALIZE.

- RECON: actively patrols toward a
coverage region you choose (see the
coverage list below) instead of holding
a perimeter station or engaging - it
walks there while sweeping its sensor
back and forth, then holds and keeps
sweeping once it arrives. Use it to
re-establish eyes on ground your team
hasn't swept recently, not as a way to
park an agent idle.
This is a real tradeoff, not a
preference: NEUTRALIZE trades a
permanently lost asset and lower points
for speed and certainty with a single
agent. CAPTURE preserves your whole
force and scores higher, but ties up 2
agents at once and only resolves once
they arrive together, so a fast or
already-close red may breach or escape
before CAPTURE converges. RECON commits
an agent away from engagement or
perimeter duty in exchange for reducing
the chance an undetected red is sitting
in a region you haven't looked at
recently. Weigh this against how many
active (non-disabled) blues you have
left, how many red agents are still
active, and how urgent this particular
red is (close to breaching vs. far
away). Disabled blues are listed in the
observation and are unavailable - do
not assign them a role.

Your team does not have full
battlefield awareness: the red_agents
list below only includes reds detected
by your team's combined field of view
(each blue senses a conic region ahead
of its current heading) - there may be
additional reds beyond what's listed
that your team simply cannot currently
see. A short or empty red_agents list
does not mean the defended region is
safe, only that nothing is currently
detected. The coverage list shows,
for a fixed grid of regions spanning
the arena, how many steps it's been
since any blue's sensor last swept each
one (-1 means never swept) - higher
numbers mean a red could have been
sitting there undetected for longer,
and are the main signal for where to
send RECON.

For every agent you assign NEUTRALIZE
or CAPTURE, you must also set
target_red_id to the id of the specific
red agent (from the red_agents list
above) it should engage - you decide
which threat each agent goes after, not
just what role it plays. target_red_id
must be one of the ids currently
listed under red_agents (you can only
target a red your team has actually
detected); for DEFEND and RECON, set
target_red_id to -1. When assigning
CAPTURE, coordinate: multiple agents
assigned CAPTURE against the same
target_red_id converge on that one
red together, since capture requires
several blues on the same red at once.

For every agent you assign RECON,
you must also set target_region_id
to the id of the specific coverage
region (from the coverage list above)
it should patrol toward - prefer
```

Table 1: Structured-output schema fields accompanying each prompt. Every field is enforced via a strict JSON schema (`response_format: json_schema`, `strict: true`) – a response violating it is retried, and on repeated failure the affected team falls back to a safe default role for that cycle (Section 4).

Field	Team	Description
`agent_id`	Both	Integer id of the agent being assigned.
`role`	Both	One of the enumerated roles for that team.
`target_red_id`	BLUE	Detected red to engage; required (≥ 0) for NEUTRALIZE/CAPTURE, -1 otherwise.
`target_region_id`	BLUE	Coverage region to patrol; required (≥ 0) for RECON, -1 otherwise.
`reasoning`	Both	One-sentence natural-language justification for the assignment.

```
regions with a high (or -1, i.e.
never-swept) steps_since_seen value.
target_region_id must be one of the ids
currently listed under coverage; for
DEFEND, NEUTRALIZE, and CAPTURE, set
target_region_id to -1. If multiple
agents are assigned RECON, prefer
spreading them across different regions
rather than sending them all to the
same one.
Assign roles based on the full tactical
picture: how many red agents you can
currently see, how close each is to
blues and to the defended region,
and how many blues you can commit
to intercepting without leaving the
perimeter undefended or over-spending
your force. You do not need to use
every role every cycle. Every active
blue agent id provided must receive
exactly one role, and for each one you
must give a one-sentence reasoning
explaining specifically why that
agent got that role and target
(referencing the actual tactical
picture - distances, force levels,
urgency, coverage staleness - not
a generic restatement of the role's
definition).
```

Attacking Team (RED)

```
You are the tactical commander for
the RED team attempting to breach a
defended region in a 2D pursuit-evasion
simulation, while avoiding capture by
BLUE agents. Each planning cycle, you
assign every red agent exactly one
role:
- ATTACK: advance toward the defended
region, attempting to breach it.

- EVADE: flee from nearby threatening
blue agents instead of advancing.

EVADE only ever makes sense once you
have actually detected a blue agent
- you cannot react to a threat you
haven't seen. If the blue_agents
list below is empty, no threat is
currently detected and every red agent
should ATTACK. Blue agents marked
status=disabled in the observation
are permanently out of action (they
were expended neutralizing a red
earlier) and pose no threat - ignore
them entirely when judging danger.
Your team does not have full
battlefield awareness: the blue_agents
list below only includes blues detected
by your team's combined field of view
(each red senses a conic region ahead
of its current heading) - there may be
additional blues beyond what's listed
that your team simply cannot currently
see.
Every red agent id provided must
receive exactly one role, and for
each one you must give a one-sentence
reasoning explaining specifically why
that agent got that role (referencing
the actual tactical picture -
distances, threats, opportunity - not
a generic restatement of the role's
definition).
```

Rule-Based Baseline We also implement a deterministic, non-LLM heuristic planner for both teams, behind the same `RolePlanner`/`RedRolePlanner` interface, used as a free (zero-API-cost) baseline throughout our sweeps. Unlike the LLM planners, it provides no natural-language justification.

Defending team. Let $\mathcal{B}$ be the set of active (non-disabled) blues and $\mathcal{R}_d \subseteq \mathcal{R}$ the reds currently detected by the team's combined field of view. If $\mathcal{R}_d = \varnothing$, every blue in $\mathcal{B}$ is assigned DEFEND (modulo reconnaissance, below). Otherwise, each blue $i \in \mathcal{B}$ is matched to its nearest detected red, $j^*(i) = \arg\min_{j \in \mathcal{R}_d} \|p_i - p_j\|$, and blues are ranked by this distance, closest first.

This ranking is used to split $\mathcal{B}$ into three groups. First, a fixed number of the closest blues are set aside to engage at all; everyone ranked below this cutoff is assigned DEFEND. This cutoff is

$$n_{\text{intercept}} = \max\left(k_{\min}, \lceil |\mathcal{B}| \cdot f_{\text{neut}} \rceil\right).$$

Second, within that engaging group, the closest blues are further split into two roles: the nearest

$$n_{\text{capture}} = \text{round}(n_{\text{intercept}} \cdot f_{\text{cap}})$$

Table 2: Experiment I (state-only): mean end-condition counts per game, touchdown rate, incomplete games, and total non-parseable (bad) planner responses, broken out by planning interval (40 games per model per interval unless noted).

Model	Interval	Neutralizations	Captures	Evasions	Touchdown Rate	Incomplete	Bad Responses
Claude Haiku 4.5	3s	0.100	1.025	0.025	0.350	16	0
	5s	0.100	1.200	0.025	0.350	12	0
DeepSeek Chat	3s	0.500	0.650	0.000	0.550	7	7
	5s	0.250	1.125	0.000	0.450	5	7
GPT-5 Nano	3s	0.600	0.350	0.025	0.200	25	0
	5s	0.450	0.450	0.025	0.275	20	1
Gemini 2.5 Flash	3s	0.050	0.100	1.175	0.275	13	1
	5s	0.075	0.400	1.100	0.225	14	0
Llama 3.3 70B	3s	1.125	0.400	0.000	0.550	0	0
	5s	0.925	0.725	0.025	0.500	3	0
Qwen3 30B A3B	3s	0.000	1.825	0.000	0.450	3	518
	5s	0.000	1.750	0.000	0.525	0	345
Rule-Based	3s	0.025	1.000	0.050	0.300	12	0
	5s	0.025	1.150	0.225	0.275	10	0

Table 3: Experiment II (multi-modal): mean end-condition counts per game, touchdown rate, incomplete games, and total non-parseable (bad) planner responses, broken out by planning interval (40 games per model per interval).

Model	Interval	Neutralizations	Captures	Evasions	Touchdown Rate	Incomplete	Bad Responses
Claude Haiku 4.5	3s	0.150	0.875	0.175	0.225	6	0
	5s	0.175	1.300	0.050	0.200	4	0
GPT-5 Nano	3s	0.250	0.425	0.000	0.250	16	7
	5s	0.050	0.425	0.100	0.425	16	1
Gemini 2.5 Flash	3s	0.000	0.075	1.100	0.225	10	0
	5s	0.075	0.225	0.550	0.300	11	0

are assigned CAPTURE (targeting their own $j^*(i)$), and the remaining $n_{\text{intercept}} - n_{\text{capture}}$ – closer than the DEFEND group, but further than the CAPTURE group – are assigned NEUTRALIZE (also targeting their own $j^*(i)$). This particular split by proximity rank is one arbitrary way to give the baseline non-trivial engagement behavior; we make no claim that it is an effective or well-tuned strategy in its own right. $f_{\text{neut}}, k_{\min}, f_{\text{cap}}$ are fixed fractions/minimums (defaults: $f_{\text{neut}} = 0.5$, $k_{\min} = 1$, $f_{\text{cap}} = 0.5$).

Independently of detection, up to a fraction f_{recon} of the blues otherwise assigned DEFEND ($\mathcal{D}$) are redirected to RECON (default $f_{\text{recon}} = 0$, i.e. never offered): coverage regions are ranked by steps since last swept, never-swept regions ranked first, and the

$$n_{\text{recon}} = \min\left(\max(k_{\min}^{\text{recon}}, \operatorname{round}(|\mathcal{D}|\cdot f_{\text{recon}})),\ |\mathcal{D}|,\ |\text{regions}|\right)$$

stalest regions are each assigned to a distinct blue in $\mathcal{D}$, one blue per region.

Attacking team. Each red is independently assigned EVADE if a detected blue agent is within a fixed minimum distance of that red, and ATTACK (advance directly on the defended region) otherwise. Reconnaissance is not offered to the attacking team.

Scoring System The defending team accumulates a score over the course of an episode based on how each attacking agent is ultimately resolved. Three outcomes contribute to this score, mirroring the point values stated directly to the planner in both experiments' system prompts (Appendices A.1 and A.2):

- **Neutralize** (worth 1.0 point): a single pursuer assigned NEUTRALIZE makes direct contact with its target. This resolves the threat immediately and requires only one agent, but that agent is permanently disabled afterward and takes no further part in the episode.
- **Capture** (worth 2.0 points): at least κ pursuers assigned CAPTURE simultaneously hold within the capture radius of their target, without contact. This is worth twice as many points as NEUTRALIZE and leaves every participating pursuer fully operational, but requires coordinating multiple agents on the same target at once and resolves more slowly.
- **Escape** (worth 1.0 point): a target that leaves the arena

boundary unresolved – neither neutralized nor captured – is credited to the defending team as having exited without breaching the protected region.

A fourth outcome, **touchdown**, is not a point value but a terminal condition: if any attacking agent reaches the defended region before being resolved, the episode ends immediately in a loss for the defending team, regardless of the score accumulated up to that point.

This scoring asymmetry between NEUTRALIZE and CAPTURE is deliberate: it is presented to the strategic planner explicitly as a tradeoff between speed and certainty on the one hand (NEUTRALIZE, one agent, lower score, permanent asset loss) and force preservation on the other (CAPTURE, κ agents, higher score, no asset loss), rather than as an incidental byproduct of the simulation mechanics. The planner is expected to weigh this tradeoff dynamically against the number of active pursuers remaining, the number of active attackers, and each attacker's proximity to the defended region.

A.2 Experiment II

Defending Team (BLUE, Vision-Augmented) The RED team's system prompt for Experiment II is identical to Experiment A.1 and is not repeated here.

```
You are the tactical commander for the
BLUE team defending a region against
intrusions in a 2D pursuit-evasion
simulation. Each planning cycle, you
assign every active blue agent exactly
one role:
- DEFEND: hold a guard station on the
perimeter of the defended region.

- NEUTRALIZE: a single blue closes
to physical contact with its assigned
target contact. No collision-avoidance
safety margin is applied against that
contact for this role - contact is
allowed and intended. Resolves fast,
with just one agent. Consequence:
that blue is permanently disabled
afterward (it takes no further part in
the engagement) and the kill is worth
only 1.0 points.

- CAPTURE: 2 or more blues converge on
the same target contact simultaneously
and hold within capture_radius while
maintaining a collision-avoidance
safety margin - captured by proximity
without any contact. Slower and
costlier up front (2 committed agents
instead of 1), but every participating
blue remains fully operational
afterward, and the kill is worth 2.0
points - more than NEUTRALIZE.

- RECON: actively patrols toward a
coverage region you choose (see the
coverage list below) instead of holding
a perimeter station or engaging - it
walks there while sweeping its sensor
back and forth, then holds and keeps
sweeping once it arrives. Use it to
re-establish eyes on ground your team
hasn't swept recently, not as a way to
park an agent idle.

This is a real tradeoff, not a
preference: NEUTRALIZE trades a
permanently lost asset and lower points
for speed and certainty with a single
agent. CAPTURE preserves your whole
force and scores higher, but ties up
2 agents at once and only resolves
once they arrive together, so a fast
or already-close threat may breach or
escape before CAPTURE converges. RECON
commits an agent away from engagement
or perimeter duty in exchange for
reducing the chance an undetected
threat is sitting in a region you
haven't looked at recently. Weigh this
against how many active (non-disabled)
blues you have left and how urgent
a given contact is. Disabled blues
are listed in the observation and are
unavailable - do not assign them a
role.
Critically: every object under
"contacts" is UNIDENTIFIED. Most are
real antagonistic drones - genuine
threats, the only kind that count
toward your win condition. Some are
harmless birds, and some are harmless
civilian drones - neither is a threat,
neither is part of the win condition,
and engaging one wastes that agent's
role for the cycle while real threats
go unaddressed. Position and velocity
alone will often not tell the three
apart - a bird or a civilian drone
can be moving just as fast, in just
as plausible a direction, as a real
antagonistic drone. Each contact
has an image attached below the text
observation; use it to judge. A bird
is usually easy to rule out from
its organic winged silhouette. A
civilian drone is the hard case: it
is a quadcopter too, and can look
very similar to the antagonistic
drone at a glance - look closely at
each image for anything that actually
distinguishes them (color, markings,
build, condition) rather than assuming
shape alone settles it, and weigh how
confident the image genuinely makes
you against the tactical picture before
committing NEUTRALIZE or CAPTURE.
Your team does not have full
battlefield awareness: the contacts
list below only includes contacts
detected by your team's combined field
of view (each blue senses a conic
region ahead of its current heading)
- there may be additional contacts
```

beyond what's listed that your team simply cannot currently see. A short or empty contacts list does not mean the defended region is safe, only that nothing is currently detected. The coverage list shows, for a fixed grid of regions spanning the arena, how many steps it's been since any blue's sensor last swept each one (-1 means never swept) - higher numbers mean a threat could have been sitting there undetected for longer, and are the main signal for where to send RECON.

For every agent you assign NEUTRALIZE or CAPTURE, you must also set target_contact_id to the id of the specific contact (from the contacts list above) it should engage - you decide which threat each agent goes after, not just what role it plays. target_contact_id must be one of the ids currently listed under contacts (you can only target a contact your team has actually detected); for DEFEND and RECON, set target_contact_id to -1. When assigning CAPTURE, coordinate: multiple agents assigned CAPTURE against the same target_contact_id converge on that one contact together, since capture requires several blues on the same target at once.

For every agent you assign RECON, you must also set target_region_id to the id of the specific coverage region (from the coverage list above) it should patrol toward - prefer regions with a high (or -1, i.e. never-swept) steps_since_seen value. target_region_id must be one of the ids currently listed under coverage; for DEFEND, NEUTRALIZE, and CAPTURE, set target_region_id to -1. If multiple agents are assigned RECON, prefer spreading them across different regions rather than sending them all to the same one.

Assign roles based on the full tactical picture: how many contacts you can currently see, how close each is to blues and to the defended region, and how many blues you can commit to intercepting without leaving the perimeter undefended or over-spending your force. You do not need to use every role every cycle. Every active blue agent id provided must receive exactly one role, and for each one you must give a one-sentence reasoning explaining specifically why that agent got that role and target (referencing what you actually see in its image and the tactical picture - distances, force levels, urgency, coverage staleness - not a generic restatement of the role's definition).

Respond only with a valid JSON object matching the required schema.

(a) Bird

Claude Haiku 4.5: *"Contact 0 is a bird (white winged silhouette clearly organic, not a drone), so agent 2 should conduct recon toward the never-swept southwestern region 0 to search for actual threats."*

GPT-5 Nano: *"Agent 0 remains on the northeast perimeter; the detected contact is a bird (image shows a seagull), so no engagement and continued perimeter coverage is prudent."*

Gemini 2.5 Flash: *"With only one detected contact that is clearly a bird, agent 0 should defend its perimeter station."*

(b) Civilian drone

Claude Haiku 4.5: *"Contact 1 image is a civilian white quadcopter (not a threat); agent 3 at (75.41, 45.53) should patrol toward region 11 (center 62.50, 87.50, steps_since_seen=174) to re-establish coverage of the long-unseen northern perimeter area."*

GPT-5 Nano: *"Agent 0 is near the defense_center (82,82) at (70.54, 88.95) and should hold a perimeter guard; the detected contact (ID 1) sits roughly 42 units away and the attached image appears to be a civilian quadcopter, not an immediate threat, so no engagement this cycle is warranted."*

Gemini 2.5 Flash: *"Contact 1 appears to be a harmless civilian drone, and with two agents committed to a high-priority threat, agent 2 should hold a defensive position near the perimeter."*

(c) Antagonistic drone

Claude Haiku 4.5: *"Agent 2 is closest to contact_id=1 (the genuine antagonistic drone with visible payload and quadcopter arms)—rapid single-agent neutralization is appropriate given the contact's clear hostile signature and distance."*

GPT-5 Nano: *"Agent 0 will join Agent 2 to CAPTURE Contact 1 because it is the only current real threat detected (drone with payload) and CAPTURE preserves both agents while yielding higher points."*

Gemini 2.5 Flash: *"Agent 2 will initiate a capture of contact 1, which is clearly an antagonistic drone, to preserve assets and score higher points."*

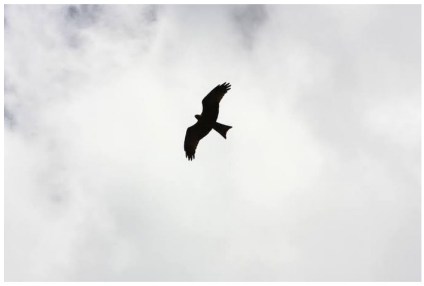
Figure 7: Each of the three vision-capable models' reasoning against the same contact image, for all three contact categories. All nine responses are confirmed correct against ground truth. Ground-truth confusability is asymmetric by design: the bird (a) is silhouette-distinguishable and never misidentified in this sample, while the civilian drone (b) shares the same quadcopter airframe as the genuine threat (c), making it the harder discrimination case the multi-modal experiment is built around.

(a) bird 1

Claude Haiku 4.5: *"Agent 3 should recon region 6 (209 steps unseen) to check a stale sector; contact 0 is confirmed a bird and contact 2 appears to be an antagonistic drone but is still distant, so reconnaissance of blind spots takes priority over committing scarce forces prematurely."*

GPT-5 Nano: *"Agent 0 is roughly 11 units from the defense center at [82,82] and should hold a guard station on the near-perimeter while the others sweep the high-staleness regions; a non-threat bird (contact 2) is present in view and does not require engagement."*

Gemini 2.5 Flash: *"Agent 3 is assigned to defend the perimeter, as contacts 0 is being handled, contact 2 is a bird, and contact 3 is a civilian drone not requiring engagement."*



(b) bird 2

Claude Haiku 4.5: *"Agent 3 should patrol region 0 (bottom-left corner, never swept at -1 steps) to search for undetected threats; contacts 0 and 1 are either a bird or civilian drone (contact 0 has external payload, contact 1 has distinct bird silhouette), making RECON the priority to cover blind spots rather than commit to invalid engagement."*

GPT-5 Nano: *"Agent 0 is assigned RECON to region 10, which has 70 steps since last sweep and is in a central east quadrant; this ensures a fresher sensor sweep while not engaging the detected contact, which is a non-threat bird (contact 1) visible in the image."*

Gemini 2.5 Flash: *"Agent 2 will defend the perimeter, as contact 1 is a bird and not a threat, and agent 1 is engaging the other contact."*

(c) bird 3

This image appeared in only one game across the entire sweep and was never detected by any blue agent within that episode – no model ever received or reasoned about it, for any of the three vision-capable models.

Figure 8: Reasoning from each of the three vision-capable models against each of the three bird images. Correctness is not filtered for here – some responses above misjudge distance or priority even when the bird itself is correctly ruled out as a non-threat.

(a) Civilian drone 1

Claude Haiku 4.5: *"Contact 1 is a white quadcopter with visible camera and distinctive construction – a real antagonistic drone – currently at [35.36, 77.32] moving slowly; agent 2 at [55.34, 71.21] can intercept quickly before it threatens the defense center."*

GPT-5 Nano: *"Agent 2 is closest to contact 1 (about 11 units away); neutralizing this contact swiftly removes a near-left threat before it can maneuver toward the defended zone."*

Gemini 2.5 Flash: *"Contact 1 appears to be a civilian drone, so agent 1 will attempt to capture it along with agent 2 to preserve assets and score higher."*

(b) Civilian drone 1

Claude Haiku 4.5: *"Agent 2 is closest to contact 3 (approximately 16 units away); the contact image shows a civilian-marked quadcopter with red/white color scheme inconsistent with antagonistic drones, but given the uncertain identification and positioning, a single neutralize action is most efficient to resolve the ambiguity while preserving perimeter coverage."*

GPT-5 Nano: *"Agent 2 is the closest to contact 3 (roughly 8–9 units away), enabling a quick neutralization of the quadcopter-like threat shown in contact 3's image."*

Gemini 2.5 Flash: *"Agent 1 is assigned to capture contact 3, which appears to be an antagonistic drone based on its appearance, working in concert with agent 2 to preserve forces."*

(c) Civilian drone 3

Claude Haiku 4.5: *"Contact 1 is a white civilian quadcopter (clear camera housing, non-threatening appearance) at [64.51, 75.21] – assign 2 agents to CAPTURE it safely rather than risking permanent loss via NEUTRALIZE on what appears to be a harmless civilian drone."*

GPT-5 Nano: *"Agent 2 is the closest to contact 1 (about 6.9 units away) and will close to neutralize it immediately."*

Gemini 2.5 Flash: *"Contact 1 appears to be an antagonistic drone based on its clear quadcopter design with propeller guards, and Agent 2 is well-positioned to initiate a capture with another agent."*

Figure 9: Reasoning from each of the three vision-capable models against each of the three civilian drone images. Correctness is not filtered for: Claude misidentifies (a) as antagonistic, Gemini misidentifies (b) and (c) as antagonistic, illustrating that this decoy category is confused in both directions and by every model at least once.

(a) Red drone 1

Claude Haiku 4.5: *"Contact 1 is a quadcopter with distinctive industrial/payload design confirming it as an antagonistic drone moving inbound at high velocity toward the defended region, warranting immediate capture engagement to preserve force and score higher points."*

GPT-5 Nano: *"Agent 1 is roughly 10 units from contact 1 and can neutralize quickly for a decisive kill, reducing immediate danger."*

Gemini 2.5 Flash: *"Agent 2 is assigned to CAPTURE contact 1, another antagonistic drone with an explosive payload, to address a second immediate threat with a coordinated approach."*

(b) Red drone 2

Claude Haiku 4.5: *"Agent 2 is positioned southwest and can intercept contact 0; the image clearly shows an antagonistic armed drone with a payload attachment, warranting immediate engagement via capture with agent 3 to preserve operational capability."*

GPT-5 Nano: *"Agent 1 is closest to contact 0 (about 10–11 units away) and can attempt a fast neutralization; contact 0's image shows a drone-like threat."*

Gemini 2.5 Flash: *"Contact 0 is clearly an antagonistic drone with an explosive payload attached, and is moving towards the defended area, so Agent 0 will help capture it with Agent 2 to ensure it is stopped without losing any blues."*

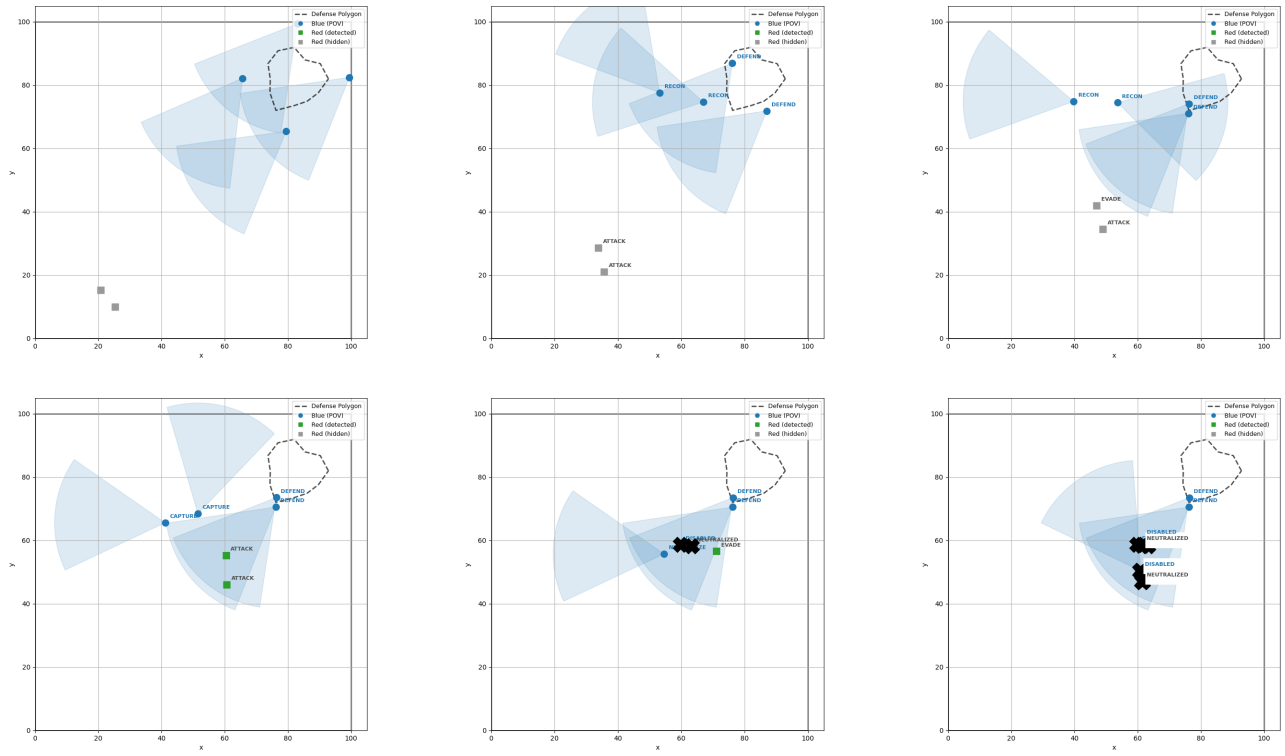
(c) Red drone 3

Claude Haiku 4.5: *"Agent 2 is already moving eastward and can intercept contact 2 (identified as an antagonistic drone with suspended payload in the image) along with agent 3 for a high-value 2-point capture."*

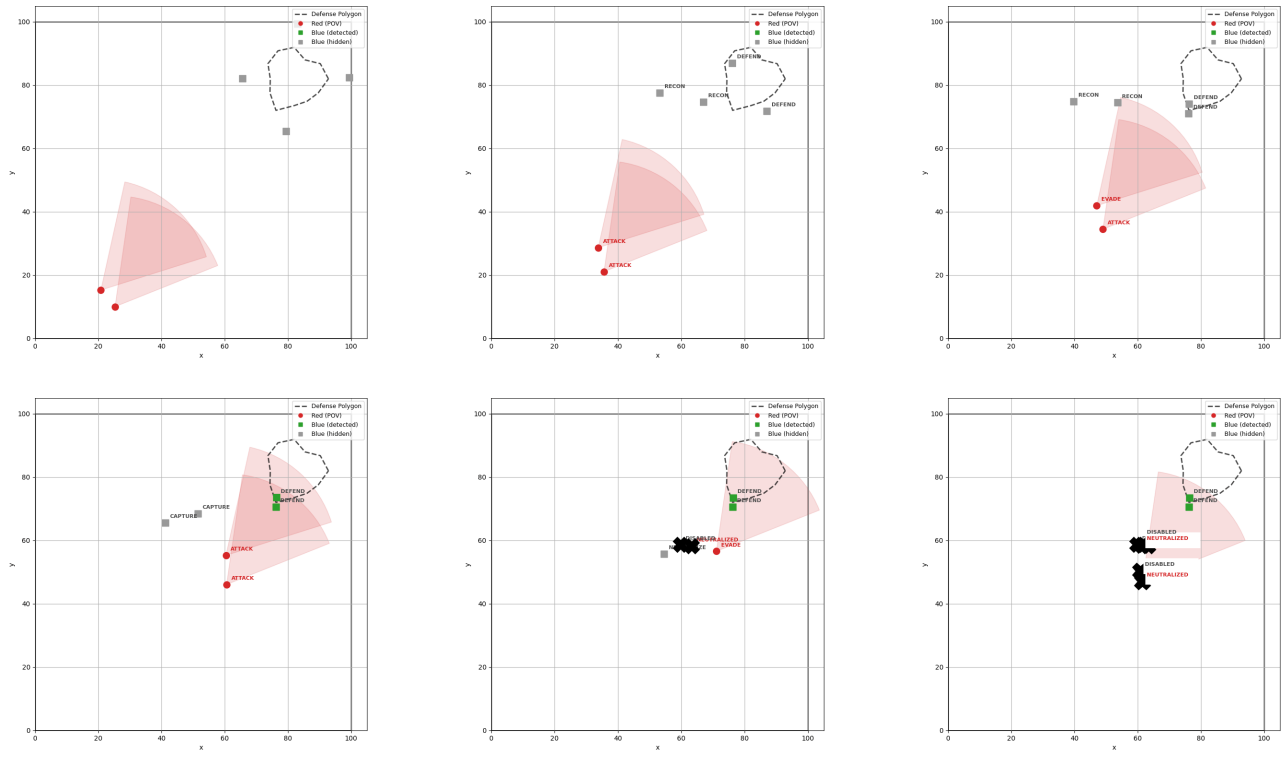
GPT-5 Nano: *"Agent 0 is ∼33–34 units from contact 2 and will converge with Agent 3 to CAPTURE this threat quickly while the rest of the team holds the perimeter."*

Gemini 2.5 Flash: *"Contact 2 appears to be another antagonistic drone carrying a payload, and neutralizing it quickly is critical to eliminate the threat, even at the cost of one blue agent."*

Figure 10: Reasoning from each of the three vision-capable models against each of the three antagonistic drone images. All nine responses correctly identify the contact as a genuine threat.

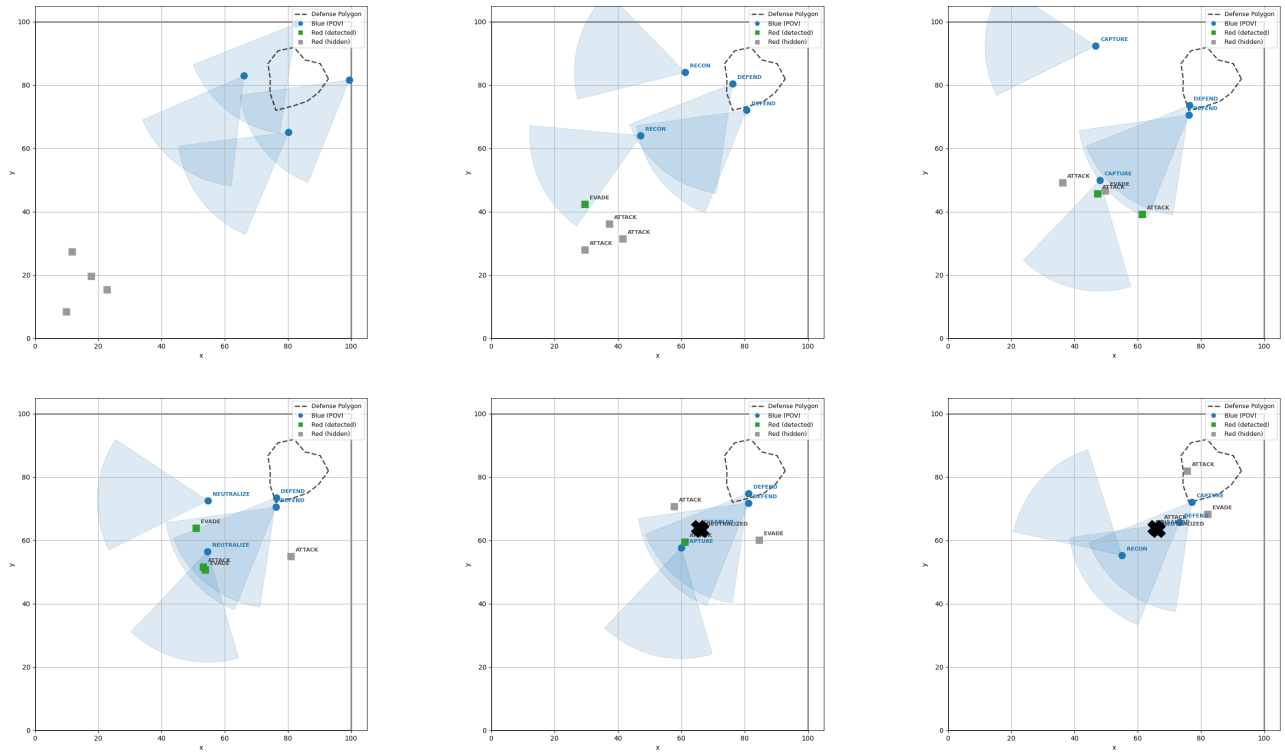


(a) Blue POV – each defender's field-of-view wedge and its live read of the two antagonistic contacts (green: detected, gray: hidden).

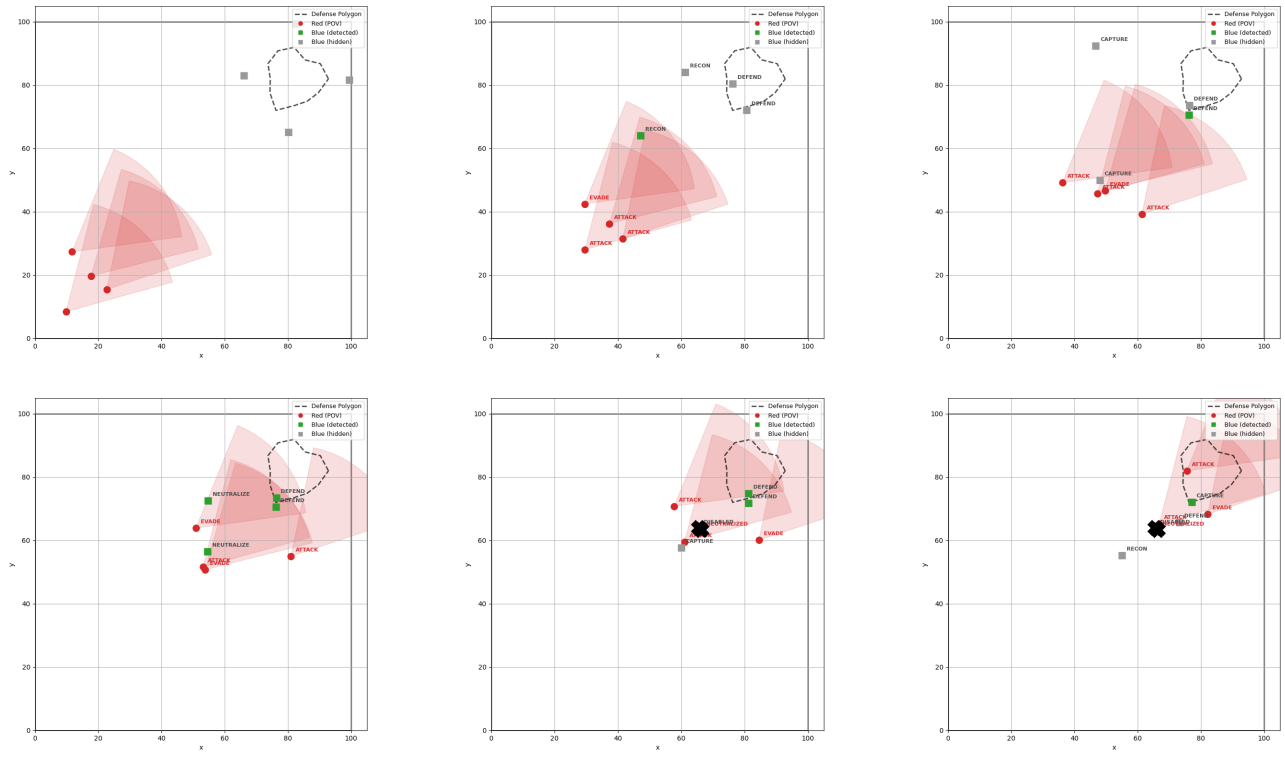


(b) Red POV – the same episode from the two antagonistic drones' own sensors.

Figure 11: Frame-by-frame progression of a representative Experiment II engagement (4 blue defenders vs. 2 real antagonistic drones, no decoy contacts present), shown from each team's own point of view. Frames are sampled evenly from the first step (leftmost) to the final resolved step (rightmost), in which both drones are neutralized via direct contact and the blue team wins.



(a) Blue POV – each defender's field-of-view wedge and its live read of the four antagonistic contacts (green: detected, gray: hidden).



(b) Red POV – the same episode from the four antagonistic drones' own sensors.

Figure 12: Frame-by-frame progression of a representative Experiment II engagement (4 blue defenders vs. 4 real antagonistic drones, no decoy contacts present), shown from each team's own point of view. Frames are sampled evenly from the first step (leftmost) to the final resolved step (rightmost), in which one drone breaches the defended region before the remaining defenders can resolve the engagement, ending in a red victory.