PURSUITVLA: AN EXECUTABLE VISION–LANGUAGE–ACTION POLICY FOR ZERO-SHOT GENERALIZATION IN PURSUIT–EVASION GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Pursuit–evasion games require pursuers to coordinate over long adversarial interactions and complex map geometry while producing executable actions. Existing graph-based methods depend on map-specific representations, while vision–language agents are predominantly designed for single-agent tasks, making coordinated and executable multi-agent action generation difficult. We introduce PursuitVLA, an autoregressive vision–language–action policy that generates reasoning traces and executable waypoint tokens from top-down observations and multi-agent states, conditioning each pursuer on previously generated teammate actions. The same role-conditioned policy also operates as an evader. Training first uses dynamic-programming (DP) demonstrations with strategic reasoning and then applies reinforcement fine-tuning with Sequential Action Compensation (SeqAC), which scores candidate actions at intermediate states induced by policy-generated teammate prefixes, allowing later agents to compensate for deviations in earlier teammate actions. Without retraining or map-specific graph reconstruction, PursuitVLA achieves 79.6% capture against DP evaders on eight unseen maps, compared with 63.0% for EPG, the strongest graph-learning baseline. It remains effective under local map perturbations and transfers from 2v1 training to simulated 10v1 pursuit. Without additional training, the same policy controls two pursuers in a closed-loop outdoor deployment, achieving an 80% capture rate (16/20 trials) against a DP-controlled evader, compared with 50% (10/20 trials) for EPG. Our project is available at: PursuitVLA.

1 INTRODUCTION

Pursuit–evasion games (PEGs) capture a broad class of adversarial decision-making problems arising in robotics, security, and autonomous systems (Vidal et al., 2001; 2002; Chung et al., 2011). The core challenge is to coordinate multiple pursuers over long horizons across varied map geometries while the evader adapts to their decisions. Search-based heuristics provide geometric baselines for this setting (Fig. 1a), but they do not represent these adaptive multi-agent dependencies.

Classical game-theoretic and dynamic-programming (DP) methods compute strong policies on discretized graph representations (Fig. 1b) (Zhou et al., 2012; Vieira et al., 2008). However, each new map requires constructing a map-specific graph and resolving the corresponding value function before deployment. Recent graph-based reinforcement-learning (RL) methods reduce repeated policy optimization by learning across graph instances (Fig. 1c) (Xue et al., 2021; 2022; Li et al., 2023; 2024); EPG, for example, transfers equilibrium policies across such instances (Lu et al., 2025). Nevertheless, these policies still require explicit graph representations or graph-based actions, so geometric changes must be encoded in the graph before the policy can respond.

Pretrained vision–language models (VLMs) offer a promising basis for policies that operate directly on visual maps by combining visual understanding with autoregressive reasoning over textual multi-agent context (Bai et al., 2025). Vision–language–action (VLA) systems further connect these representations to action generation (Zitkovich et al., 2023; Zhou et al., 2025; Chen et al., 2025). Applying these capabilities to adversarial multi-agent pursuit–evasion, however, presents two challenges (Fig. 1d). *1) Teammate-conditioned coordination.* Existing systems predominantly

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

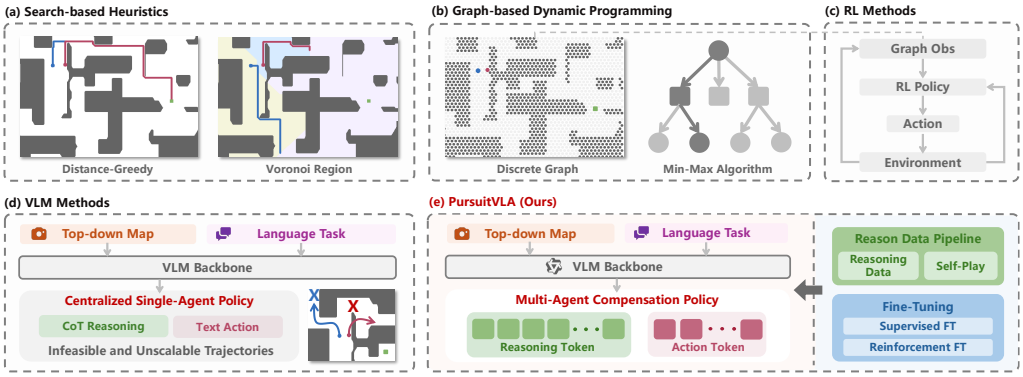


Figure 1: **Comparison of pursuit–evasion paradigms.** Unlike prior methods, PursuitVLA integrates visual map understanding with interpretable reasoning and generates coordinated multi-agent actions through a unified compensation policy.

target single-agent control and do not directly model how one agent's action changes the appropriate decisions for its teammates. *2) Executable action generation.* Language or symbolic action outputs require additional grounding before they can be converted into executable waypoints.

To address these challenges, we propose PursuitVLA, an autoregressive VLA policy for cross-map zero-shot pursuit–evasion (Fig. 1e). Given a top-down visual observation and the current multi-agent state, PursuitVLA sequentially generates dedicated waypoint tokens conditioned on previously assigned teammate actions and then executes the resulting joint action synchronously. We first use supervised fine-tuning (SFT) on equilibrium trajectories paired with structured reasoning to initialize visual-to-action generation. We then combine group relative policy optimization (GRPO) (Shao et al., 2024) with Sequential Action Compensation (SeqAC) (Fig. 2d). SeqAC evaluates candidate actions from intermediate states induced by policy-generated teammate prefixes, assigning credit to later-pursuer actions that compensate for earlier deviations rather than treating the expert continuation conditioned on the original teammate actions as the target.

Our contributions are summarized as follows:

- **Executable autoregressive VLA policy.** We introduce PursuitVLA, the first VLA policy for pursuit–evasion that directly generates executable waypoint actions and coordinates multiple pursuers through autoregressive teammate conditioning.
- **Sequential Action Compensation.** We propose SeqAC, a reinforcement fine-tuning mechanism that evaluates candidate actions at intermediate states induced by policy-generated teammate prefixes and trains later pursuers to compensate for deviations in earlier teammate actions.
- **Cross-map and team-size generalization.** We demonstrate zero-shot generalization on eight unseen maps without map-specific graph reconstruction, robustness to local map perturbations, and direct team-size transfer from 2v1 training to simulated 10v1 pursuit.
- **Direct simulation-to-robot transfer.** We deploy the simulation-trained PursuitVLA without additional policy training in a closed-loop outdoor system comprising three Unitree Go2-W robots. Across four initial configurations, the policy controls two pursuers against a DP-controlled evader and achieves 16/20 captures (80%), compared with 10/20 (50%) for EPG under the same conditions.

2 Related Work

Pursuit–evasion games. Geometric planners, including distance-greedy and Voronoi-guided pursuit, provide efficient baselines for PEGs (Zhou et al., 2016; Pan et al., 2012). Game-theoretic and dynamic-programming methods provide theoretically grounded policies for a specified game representation (Li et al., 2024; Vieira et al., 2008), but changes in map geometry can require reconstructing the graph or state representation and resolving the corresponding policy. Recent graph-based RL methods reduce repeated policy optimization across graph-structured instances (Xue et al., 2021; 2022; Li et al., 2023; 2024); EPG, for example, uses DP-computed equilibrium policies as supervision for cross-graph transfer (Lu et al., 2025). However, these methods still require an explicit graph representation of each scene, limiting direct adaptation to the current visual geometry when the

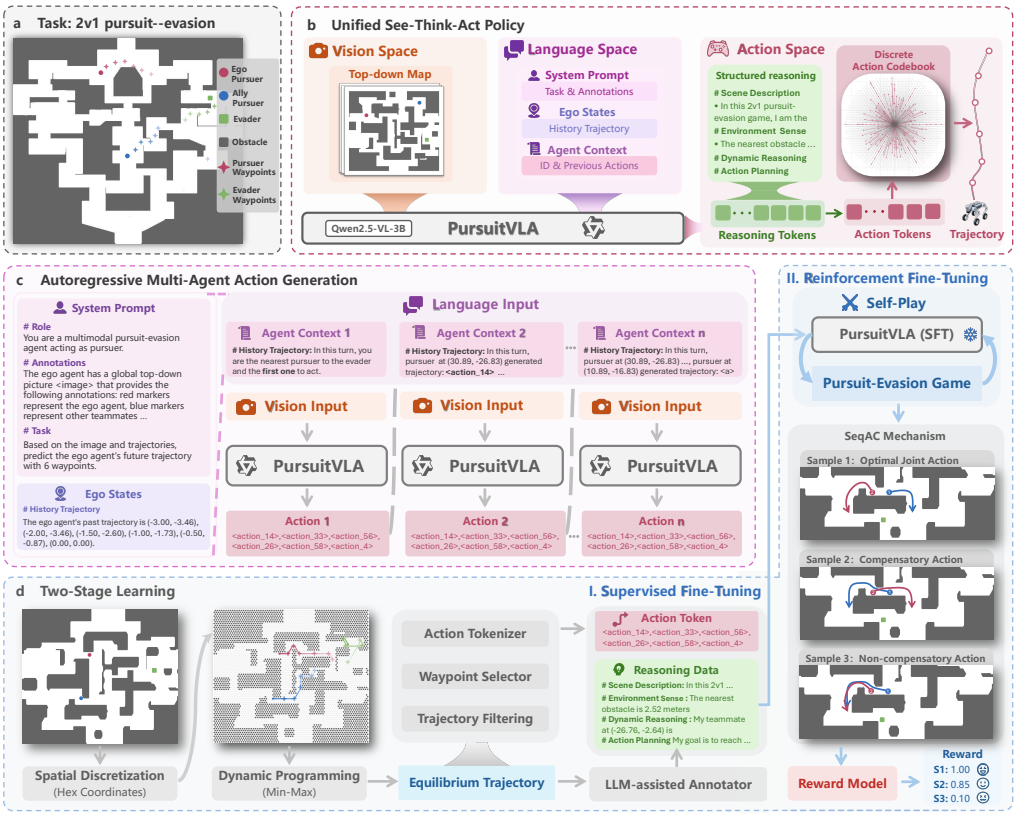


Figure 2: **Overview of PursuitVLA. a**, The 2v1 top-down pursuit–evasion setting. **b**, The see–think–act policy maps visual geometry, the multi-agent state, and teammate context to reasoning and waypoint action tokens. **c**, Autoregressive joint-action generation conditions each later pursuer decision on actions already generated for earlier teammates. **d**, Two-stage training: SFT learns from DP-generated trajectories with reasoning annotations. Self-play with the frozen SFT policy generates teammate-action prefixes, after which SeqAC scores candidates at the induced states for RFT.

layout changes. PursuitVLA instead conditions directly on the current top-down visual geometry and multi-agent state to generate coordinated waypoint actions in continuous environments.

VLM- and VLA-based agents. VLM- and LLM-based agents have been used for decision making through prompting, supervised fine-tuning, and reinforcement post-training (Yao et al., 2023; Wang et al., 2024; Zhai et al., 2024; Xu et al., 2025). Because their outputs are often expressed as language or symbolic actions, additional interfaces may be required to ground them into spatially executable controls. VLA systems couple visual perception with action-token prediction and have been applied mainly to single-agent driving and manipulation (Zhong et al., 2025; Zitkovich et al., 2023; Zhou et al., 2025; Chen et al., 2025). Compared with typical single-agent VLA settings, pursuit–evasion imposes additional requirements: actions must remain effective and executable over long adversarial interactions, while decisions must be coordinated across multiple agents. PursuitVLA addresses these requirements through waypoint action tokens, autoregressive teammate conditioning, and SeqAC-based reinforcement fine-tuning under policy-generated teammate prefixes.

3 PursuitVLA

Figure 2 overviews PursuitVLA. A pretrained VLM and unified autoregressive model map visual observations and multi-agent context to reasoning and dedicated waypoint tokens (Fig. 2b,c). The tokens decode into executable waypoint displacements, while each later pursuer conditions on actions already generated for earlier teammates. Training proceeds in two stages (Fig. 2d): SFT on expert trajectories with reasoning annotations initializes visual-to-action generation, and RFT with GRPO and SeqAC optimizes coordination under policy-generated teammate prefixes.

3.1 FRAMEWORK

Model Inputs and Autoregressive Policy. PursuitVLA takes as input a global top-down map M, a task instruction L, the ego state S_i of pursuer i, and an accumulated action context $C_{1:i-1}$ containing action tokens generated for earlier pursuers in the same step. These components and the pursuer index form the multimodal query $q_i = (M, L, S_i, C_{1:i-1}, i)$. Appendix A provides the complete input serialization and prompt format. At each timestep, pursuers are ordered by their distance to the evader. Conditioned on q_i, the model generates a reasoning segment over scene geometry and agent interactions, followed by a six-token waypoint plan $\boldsymbol{a}_i$. The plan is appended to the shared context before generating the next pursuer's decision. After all plans are generated sequentially, their first action tokens are executed synchronously, and the policy replans at the next timestep (Fig. 2c).

Action Tokenization. We represent agent actions using waypoint displacements in continuous space, rather than symbolic grid moves, allowing the model to operate in a geometrically meaningful motion space. To enable trajectory planning within the language model, continuous demonstration trajectories are first represented as sequences of waypoints. For each pair of consecutive waypoints (p_i, p_{i+1}), we compute the relative displacement $(\Delta x_i, \Delta y_i)$, yielding a local motion representation. These displacement vectors are then filtered according to their frequencies in the training trajectories to form a compact action codebook $\mathcal{A} = \{a_1, \ldots, a_K\}$ with $K = 184$ representative motions, each added to the model vocabulary as a waypoint action token. As a result, continuous trajectories can be discretized into sequences of action tokens $\boldsymbol{a} = [a_1, \ldots, a_T]$, where each token $a_t \in \mathcal{A}$ encodes a local waypoint displacement, and T is the length of the tokenized predicted trajectory. This formulation enables the VLA to perform trajectory generation as a next-token prediction problem, and during inference, the model autoregressively predicts action tokens that are decoded via the codebook into an executable waypoint trajectory. Further details are provided in Appendix G.1.

3.2 DATA PIPELINE

Training PursuitVLA uses high-quality expert trajectories paired with reasoning supervision to establish alignment between perception, reasoning, and action generation. In PEGs, acquiring such data poses two challenges: (i) obtaining expert trajectories in continuous environments consistent with multi-agent equilibrium strategies, and (ii) generating strategy-consistent reasoning explanations for these trajectories. We address these challenges via an automated data pipeline that converts graph-based equilibrium strategies into filtered continuous trajectories and aligned reasoning traces.

Figure 2d summarizes the pipeline. We first discretize each top-down map into a hex-grid graph and compute equilibrium strategies using the DP procedure from EPG (Lu et al., 2025). We then filter grid-induced artifacts and convert the resulting paths into continuous waypoint sequences mapped to the action codebook $\mathcal{A}$. A hybrid rule-based and Qwen2.5-VL-72B annotator (Bai et al., 2025) combines scene context with the filtered trajectories to generate structured, trajectory-consistent reasoning annotations. The pipeline produces 398,685 annotated training examples spanning pursuer and evader roles across multiple maps. See Appendix G for data pipeline details.

3.3 SUPERVISED FINE-TUNING

PursuitVLA is initialized from the pretrained Qwen2.5-VL-3B backbone (Bai et al., 2025) and supervised to generate reasoning tokens followed by waypoint action tokens under the input–output format defined in Sec. 3.1. For each decision step and agent i, the model is conditioned on the input $(M, L, S_i, C^{\text{gt}}_{1:i-1}, i)$ and trained to generate an output token sequence consisting of reasoning tokens followed by waypoint action tokens.

The corresponding target output is a token sequence $\boldsymbol{x}_i = [\boldsymbol{l}_i, \boldsymbol{a}_i] = [l_{i,1}, \ldots, l_{i,L_i}, a_{i,1}, \ldots, a_{i,T_i}]$, where $\boldsymbol{l}_i$ and $\boldsymbol{a}_i$ denote the reasoning and physical action token sequences, respectively. Both sequences are produced by the data pipeline in Sec. 3.2 and form a paired, trajectory-consistent supervision signal. To explicitly structure generation, we enforce a fixed output format in which reasoning tokens $\boldsymbol{l}_i$ are enclosed within `<think>`$\cdots$`</think>`, followed by action tokens $\boldsymbol{a}_i$ enclosed within `<answer>`$\cdots$`</answer>`.

In SFT, each pursuer is trained to generate its own sequence while conditioning on an autoregressive context derived from prior pursuers. Here, the autoregressive pursuer context $C^{\text{gt}}_{1:i-1}$ is composed

of ground-truth action labels obtained from the data pipeline, rather than actions predicted by the model, ensuring that the conditioning context remains accurate during training. The SFT objective is a standard causal language modeling loss:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{T}\sum_{t=1}^{T} \log \pi_\theta\big(x_t \mid x_{<t}, M, L, S_i, C^{\text{gt}}_{1:i-1}, i\big). \tag{1}$$

where T is the total number of tokens for a given agent's sequence, and π_θ is the PursuitVLA model parameterized by θ. This loss function teaches the model to (i) produce coherent reasoning, (ii) generate physically grounded actions, and (iii) follow the multi-agent autoregressive structure, where each agent's decisions are conditioned on their own context and the actions of previous agents. This provides a strong initialization for subsequent reinforcement fine-tuning.

3.4 Reinforcement Fine-Tuning with Sequential Action Compensation

Autoregressive joint-action generation allows each later pursuer to adapt its decision to previously generated teammate actions. During SFT, however, later pursuers observe only expert-consistent teammate prefixes. When an earlier action deviates from the expert trajectory, the original expert continuation may no longer be appropriate for the resulting context. Sequential Action Compensation (SeqAC) addresses this mismatch by evaluating later-pursuer actions from the intermediate states induced by policy-generated teammate prefixes.

We first collect self-play trajectories on the training maps using a frozen copy of the SFT policy for both pursuer and evader roles. The resulting states form RFT queries, where $C_{1:i-1}$ contains actions sampled for preceding pursuers rather than expert actions. This exposes later pursuers to the policy-generated teammate prefixes encountered during inference. Starting from the SFT policy, we then perform reinforcement fine-tuning using group relative policy optimization (GRPO) (Shao et al., 2024). Appendix I provides the self-play construction and optimization details.

For a joint decision at global state s_t, let $a_{<i} = (a_1, \ldots, a_{i-1})$ denote the generated teammate-action prefix. Starting from $s_t^{(1)} = s_t$, SeqAC constructs the intermediate state for pursuer i as $s_t^{(k+1)} = p_k(s_t^{(k)}, a_k)$ for $k = 1, \ldots, i-1$, where p_k updates only pursuer k and leaves all other agents unchanged. These intermediate states are used only for candidate scoring. At execution time, PursuitVLA first generates the complete joint action and then executes all pursuer actions synchronously.

For each training map, we compute and cache a DP value function $V_{\text{DP}} : \mathcal{S} \to \mathbb{R}$, where $V_{\text{DP}}(s)$ is the minimax number of decision steps remaining to capture; capture states have value zero, lower values indicate faster predicted capture, and continuous states are projected to the nearest valid DP cell before lookup. For intermediate state $s_t^{(i)}$, the lowest successor value attainable by pursuer i is

$$V_i^\star = \min_{a \in \mathcal{A}_i(s_t^{(i)})} V_{\text{DP}}\left(p_i(s_t^{(i)}, a)\right). \tag{2}$$

SeqAC scores each candidate according to its additional predicted capture steps relative to this best available continuation:

$$r^{i,j}_{\text{task}} = 1 - \frac{\min\left\{V_{\text{DP}}\left(p_i(s_t^{(i)}, a_i^j)\right), H\right\} - V_i^\star}{H}, \tag{3}$$

A candidate attaining $V_i^\star$ receives the maximum task reward, and $H = 200$ is the episode horizon used to normalize and bound the additional predicted capture steps. After a prefix deviation, the same comparison evaluates candidate actions against the best recovery available from the resulting intermediate state. The final response reward is $r_i^j = 0.9 r^{i,j}_{\text{task}} + 0.1 r^{i,j}_{\text{format}}$, where $r^{i,j}_{\text{format}} \in \{0, 1\}$ indicates whether the response follows the required output format.

Each sampled response contains a reasoning trace followed by six waypoint action tokens. Because only the first token is executed before replanning, SeqAC evaluates only this first action. Candidate rewards are normalized within each group and optimized using the token-level GRPO objective in Appendix I. The cached DP values are queried only during training and are not used on unseen maps or during robot deployment.

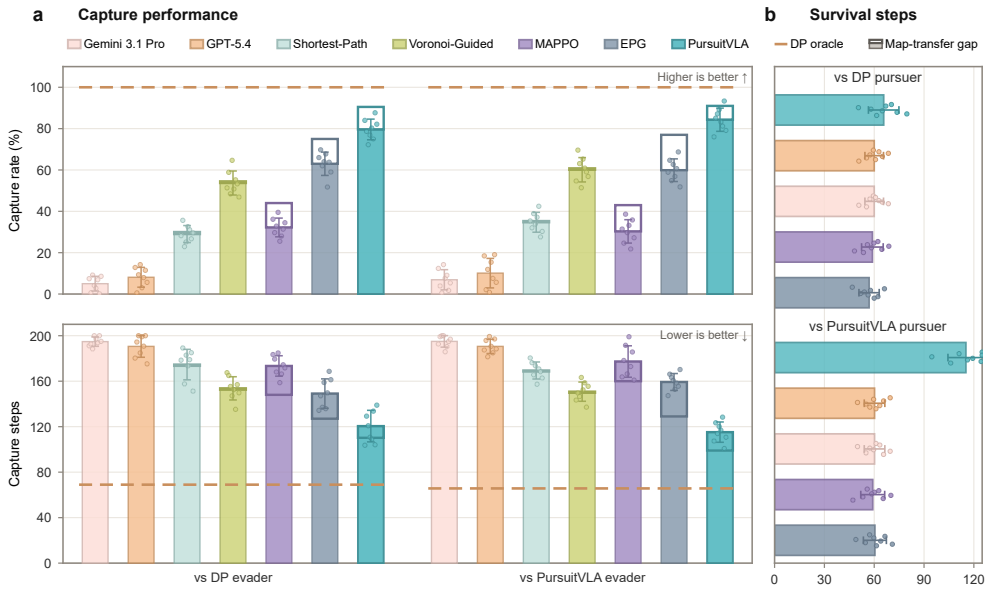


Figure 3: **Cross-map zero-shot generalization.** **a**, Pursuer capture rate and mean capture steps against DP and PursuitVLA evaders on eight unseen maps. Filled bars are unseen-map means, points are per-map means, error bars are standard deviations across maps, open outlines are training-map means, and dashed lines are map-specific DP results after recomputation. **b**, Evader survival against DP and PursuitVLA pursuers.

4 Experiments

We evaluate PursuitVLA in simulation and in a closed-loop outdoor robotic system. Simulation experiments assess cross-map generalization, robustness to map perturbations, and coordination across team sizes. The outdoor deployment tests whether the same policy can support coordinated physical pursuit without additional training when integrated with localization and navigation.

4.1 Experimental Setup

Environment. Following pursuit–evasion evaluation settings of prior work (Li et al., 2024; Lu et al., 2025), we use global top-down maps with complete multi-agent state information to study coordination and map transfer rather than exploration under partial observability. The training and unseen sets contain 12 maps adapted from Valorant and eight maps adapted from CS:GO, respectively, with obstacle-dense rooms, corridors, and narrow passages. Maps are top-down traversability images, and agents move in continuous coordinates. The primary task is 2v1 pursuit–evasion, and each episode terminates upon capture or after 200 decision steps. All map layouts are shown in Appendix B.

Evaluation settings. We evaluate PursuitVLA without further training on the eight unseen maps, locally perturbed versions of eight training maps, and larger pursuer teams than seen during training. The unseen maps are excluded from all training and model-selection stages, and all compared methods use 100 matched initial conditions per map. Capture rate is the fraction of trials ending in capture within 200 steps, while capture steps are averaged over all trials with the 200-step horizon assigned to uncaptured episodes. For evader evaluation, survival steps denote the number of steps before capture, or 200 for an escaped episode, and escape rate is the fraction of episodes without capture.

Baselines. We compare with Shortest Path (Hart et al., 1968), Voronoi-guided pursuit (Zhou et al., 2016), DP (Vieira et al., 2008), EPG (Lu et al., 2025), and MAPPO (Yu et al., 2022). We also evaluate prompted Gemini 3.1 Pro (Google DeepMind, 2026) and GPT-5.4 (OpenAI, 2026) using the same visual observation and state description. DP is recomputed on each unseen map and serves as a map-specific oracle reference. All methods share the same continuous simulator and low-level controller. Baseline details are provided in Appendix J.

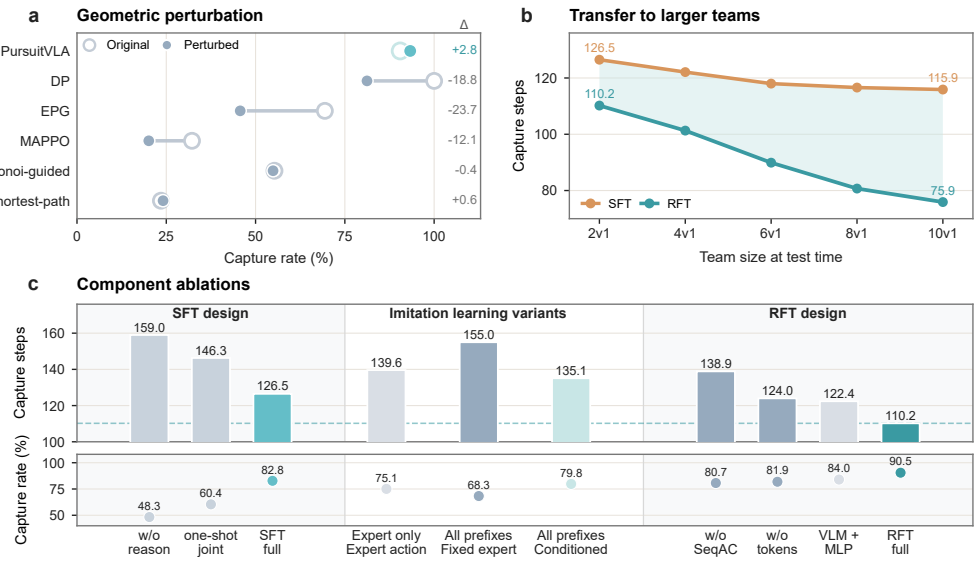


Figure 4: **Robustness and coordination analyses. a**, Capture rates before and after local map perturbations. **b**, SFT and RFT capture steps when policies trained in 2v1 are applied to teams of 2–10 pursuers. **c**, Component ablations across SFT, imitation-learning, and RFT designs.

4.2 Performance and Cross-Map Zero-Shot Generalization

Performance on training maps. On the 12 training maps, PursuitVLA achieves the highest capture rate and the lowest mean capture steps among non-oracle policies under both evaluated opponent settings (Fig. 3a; Appendix Table 6(a)). DP is reported separately as a map-specific oracle because it derives its policy from a discretized graph constructed for each map. PursuitVLA reaches 90.5% capture in 110.2 mean steps against DP evaders and 91.0% capture in 99.0 mean steps against PursuitVLA evaders.

Zero-shot transfer to unseen maps. Without retraining or map-specific recomputation, PursuitVLA achieves the highest capture rate and the lowest mean capture steps among the non-oracle methods on the eight unseen maps (Fig. 3a; Appendix Table 6(b)). Against DP evaders, it reaches 79.6% capture in 120.5 mean steps, compared with 63.0% and 149.1 steps for EPG. Against PursuitVLA evaders, it achieves 84.3% capture in 115.2 mean steps. PursuitVLA also achieves a higher capture rate than MAPPO, the geometric planners, and the prompted multimodal models on each unseen map. The recomputed DP oracle achieves 100% capture on every unseen map, serving as a map-specific reference rather than a zero-shot policy.

A unified policy for both roles. The same role-conditioned PursuitVLA model used for pursuit also acts as an evader on the eight unseen maps, without requiring a separately trained evader model (Fig. 3b; Appendix Table 6(c)). It achieves the longest mean survival among the non-oracle evader policies, surviving for 65.7 steps against the DP pursuer and 115.2 steps against the PursuitVLA pursuer. These results show that one PursuitVLA model generates executable waypoint actions for both pursuit and evasion across unseen layouts. By contrast, EPG uses separately trained role-specific policies (Lu et al., 2025), and the evaluated MAPPO baseline likewise uses separately trained pursuer and evader policies (Appendix J).

4.3 Robustness and Team-Size Transfer

Local map perturbations. We evaluate robustness by locally adding or removing obstacles on eight training maps, altering local connectivity while preserving the overall map structure. PursuitVLA maintains its capture performance under these perturbations, achieving 93.3% capture compared with 90.5% on the original maps (Fig. 4a; Appendix Table 7(a)). Under the same no-reconstruction protocol, DP, EPG, and MAPPO decrease from 100.0% to 81.2%, from 69.4% to 45.7%, and from 32.2% to 20.1%, respectively. Voronoi-guided and Shortest Path planners remain relatively stable

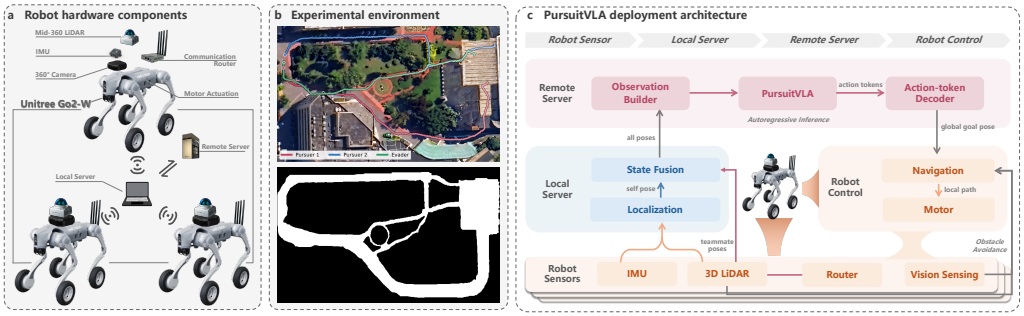


Figure 5: **Closed-loop outdoor robotic pursuit. a**, Three Unitree Go2-W robots configured as two pursuers and one DP-controlled evader. **b**, The approximately $120\,\mathrm{m} \times 60\,\mathrm{m}$ pre-mapped campus site. **c**, Deployment stack: localization and state fusion construct the global observation; PursuitVLA generates teammate-conditioned plans; the first token of each plan is decoded into a global navigation goal; updated poses close the 2 Hz control loop.

because they operate directly on the current occupancy map, but achieve only 54.9% and 24.1% capture on the perturbed maps, respectively. These results demonstrate that PursuitVLA maintains robust performance under local geometric changes without map-specific reconstruction or policy adaptation, whereas graph-based methods that rely on precomputed representations are more sensitive to changes in map connectivity.

Transfer to larger teams. The same policies trained in 2v1 are applied to 4v1, 6v1, 8v1, and 10v1 without retraining or changing the action vocabulary. RFT mean capture steps decrease from 110.2 in 2v1 to 75.9 in 10v1, while SFT decreases from 126.5 to 115.9 and largely plateaus after six pursuers (Fig. 4b; Appendix Table 7(b)). The widening gap shows that applying a policy to larger teams does not by itself ensure that additional pursuers improve capture efficiency: RFT continues to reduce capture time as the team grows, whereas SFT gains little beyond six pursuers. These results show that PursuitVLA transfers to larger teams without additional training and that RFT makes more effective use of added pursuers than SFT.

4.4 Ablation Studies

Reasoning supervision and autoregressive teammate conditioning. In the *SFT design* group of Fig. 4c and Appendix Table 7(c), full SFT achieves 82.8% capture in 126.5 mean steps. Removing reasoning annotations while retaining the same inputs and waypoint action targets lowers capture to 48.3% and increases the mean to 159.0 steps. Replacing autoregressive teammate conditioning with a one-shot variant that generates all pursuer actions in a single response from the same map and multi-agent state lowers capture to 60.4% and increases the mean to 146.3 steps. These results show that both reasoning-labelled supervision and conditioning later pursuer decisions on previously generated teammate actions contribute to the performance of the full SFT policy.

Imitation learning under teammate-prefix deviations. The *Imitation learning variants* group of Fig. 4c compares three supervised strategies for handling teammate-prefix deviations. Expert-only imitation trains on expert actions under expert-consistent prefixes and obtains 75.1% capture. When training includes deviated prefixes but retains the original expert action as the target, capture decreases to 68.3%. Recomputing the target as the DP-optimal action for each deviation-induced intermediate state improves capture to 79.8%. Although this conditioned variant is the strongest imitation strategy, it remains below full RFT with SeqAC, which reaches 90.5% capture. Thus, supervised imitation benefits from prefix-conditioned compensatory targets, but does not reproduce the improvement obtained by SeqAC under policy-generated teammate prefixes.

SeqAC, action tokens, and direct decoding. The *RFT design* group of Fig. 4c compares SeqAC, waypoint tokens, and direct decoding. In RFT without SeqAC, intermediate-state candidate scoring is replaced by candidate scoring directly from the original global state using the same DP reward, with all other RFT settings unchanged (Appendix I.1). Rather than improving on SFT, this variant performs worse than the full SFT model (80.7% capture; 138.9 vs. 126.5 steps), showing that RFT with original-state candidate scoring provides no performance gain over SFT under policy-generated

teammate prefixes. Replacing the autoregressive waypoint-token decoder with an MLP head (*VLM + MLP*) yields 84.0% capture and 122.4 steps, compared with 110.2 steps for full PursuitVLA. Without dedicated waypoint action tokens, RFT reaches 81.9% capture in 124.0 steps, yielding a lower capture rate than full SFT despite a modest reduction in capture steps. Full RFT combining SeqAC with dedicated waypoint action tokens achieves 90.5% capture in 110.2 steps, outperforming all RFT variants, full SFT, and all imitation-learning variants. These results show that applying RFT with the DP value-based reward is not sufficient by itself to improve over supervised imitation: the gain of full RFT depends on prefix-induced intermediate-state evaluation, while autoregressive waypoint-token decoding provides additional performance beyond direct VLM + MLP action prediction.

4.5 Outdoor Robotic Deployment

We deploy the simulation-trained PursuitVLA without further training in a 2v1 outdoor pursuit task using three Unitree Go2-W robots on a pre-mapped $120\,\mathrm{m} \times 60\,\mathrm{m}$ campus unseen during training (Fig. 5a–c). Two pursuers are controlled by the same PursuitVLA policy, while the third follows a DP evasion policy computed for the site.

Table 1: Outdoor deployment results.

	PursuitVLA		EPG	
Configuration	Captures	Time (s)	Captures	Time (s)
Encirclement	5/5	55.7	5/5	68.5
Same-side	4/5	208.2	3/5	237.4
Narrow entrance	4/5	247.3	1/5	287.1
Opposite-side	3/5	291.0	1/5	296.8
Overall	**16/20**	**208.7**	**10/20**	**231.9**

At each 2 Hz update, the policy server fuses robot poses with the top-down map and autoregressively generates two six-token pursuer plans, with the second plan conditioned on the previously generated plan for the first pursuer. The first waypoint token of each plan is dispatched synchronously as a global navigation goal and executed by Nav2 (Macenski et al., 2020). Robot-side visual sensing is used only for local obstacle avoidance, and the updated poses form the next policy input, closing the physical control loop. See Appendix K for deployment details.

We conduct five trials under each of four initial configurations: encirclement, same-side, narrow-entrance, and opposite-side starts. A capture occurs when the evader enters a 2 m radius around either pursuer within 5 min. PursuitVLA achieves 16 captures in 20 trials (80%), compared with 10 captures (50%) by EPG, while reducing the mean trial duration from 231.9 s to 208.7 s (Table 1). Notably, PursuitVLA achieves higher capture rates in the narrow-entrance (4/5 vs. 1/5) and opposite-side (3/5 vs. 1/5) configurations. Representative trajectories are provided in Appendix Fig. 17. These results demonstrate direct simulation-to-robot transfer of teammate-conditioned global goals for coordinated pursuit across different initial geometries.

5 Limitations and Future Work

By transferring a simulation-trained policy to outdoor robots without further training, PursuitVLA provides a concrete step toward learned policies for physical pursuit–evasion. The policy assumes access to a top-down traversability map and global multi-agent state, limiting direct operation in unmapped or partially observable environments. Training currently relies on DP demonstrations and DP-derived values on the training maps; PursuitVLA itself does not query the DP solver at inference or reconstruct graphs on unseen maps. Because sparse outcome rewards make credit assignment difficult over long horizons (Feng et al., 2025; Wei et al., 2026), extending SeqAC to domains without tractable solvers remains challenging. Future work will investigate learned critics and outcome-based evaluators that provide informative and stable action-level credit without exact solvers.

6 Conclusion

We presented PursuitVLA, an autoregressive VLA policy that generates waypoint action tokens and conditions each pursuer on actions already generated for earlier teammates. SeqAC trains these decisions under policy-generated teammate prefixes. On eight unseen maps against DP evaders, PursuitVLA improves capture rate by 16.6 percentage points and reduces mean capture steps by 19.2% over EPG, the strongest graph-learning baseline. The 2v1-trained policy extends to 10v1 without retraining and remains effective under local map perturbations. In a closed-loop outdoor deployment, the same policy controls two pursuers without additional training and achieves 16/20 captures (80%) against a DP-controlled evader, versus 10/20 (50%) for EPG.

AI USE STATEMENT

In this work, we have not used generative AI tools for tasks with required disclosure, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to improve the readability of the manuscript and assist with software coding. We have reviewed all AI-assisted work. AI-assisted text and code were reviewed and verified by both authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. In *Proceedings of Robotics: Science and Systems, RSS 2025, Los Angeles, CA, USA, Jun 21-25, 2025*, 2025. doi: 10.15607/RSS.2025.XXI.019.

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

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.10978.

Google DeepMind. Gemini 3.1 pro model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro.

Daniel Harabor and Alban Grastien. Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pp. 1114–1119, 2011. doi: 10.1609/aaai.v25i1.7994.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

Pengdeng Li, Shuxin Li, Xinrun Wang, Jakub Cerny, Youzhi Zhang, Stephen McAleer, Hau Chan, and Bo An. Grasper: A generalist pursuer for pursuit-evasion problems. *arXiv preprint arXiv:2404.12626*, 2024.

Shuxin Li, Xinrun Wang, Youzhi Zhang, Wanqi Xue, Jakub Černý, and Bo An. Solving large-scale pursuit-evasion games using pre-trained strategies. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, volume 37, pp. 11586–11594, 2023. doi: 10.1609/aaai.v37i10.26369.

Runyu Lu, Peng Zhang, Ruochuan Shi, Yuanheng Zhu, Dongbin Zhao, Yang Liu, Dong Wang, and Cesare Alippi. Equilibrium policy generalization: A reinforcement learning framework for cross-graph zero-shot generalization in pursuit-evasion games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2718–2725, 2020. doi: 10.1109/IROS45743.2020.9341207.

OpenAI. GPT-5.4 model. OpenAI API documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.4.

OpenDILab. DI-engine: Opendilab decision ai engine, version 0.5.3. Software, 2023. URL https://github.com/opendilab/DI-engine/tree/v0.5.3.

Selina Pan, Haomiao Huang, Jerry Ding, Wei Zhang, Dušan M. Stipano vić, and Claire J. Tomlin. Pursuit, evasion and defense in the plane. In *2012 American Control Conference (ACC)*, pp. 4167–4173, 2012. doi: 10.1109/ACC.2012.6315389.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

R. Vidal, S. Rashid, C. Sharp, O. Shakernia, Jin Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pp. 2948–2955 vol.3, 2001. doi: 10.1109/ROBOT.2001.933069.

R. Vidal, O. Shakernia, H.J. Kim, D.H. Shim, and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, 2002. doi: 10.1109/TRA.2002.804040.

Marcos AM Vieira, Ramesh Govindan, and Gaurav S Sukhatme. Optimal policy in discrete pursuit-evasion games. *Department of Computer Science, University of Southern California, Tech. Rep*, pp. 08–900, 2008.

Taowen Wang, Yiyang Liu, James Liang, Junhan Zhao, Yiming Cui, Yuning Mao, Shaoliang Nie, Jiahao Liu, Fuli Feng, Zenglin Xu, et al. M2pt: Multimodal prompt tuning for zero-shot instruction learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3723–3740, 2024.

Tong Wei, Yijun Yang, Changhao Zhang, Junliang Xing, Yuanchun Shi, Zongqing Lu, and Deheng Ye. GTR-Turbo: Merged checkpoint is secretly a free teacher for agentic VLM training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2512.13043.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2): 344–357, 2017. doi: 10.2514/1.G001921.

Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. Dipllm: Fine-tuning LLM for strategic decision-making in diplomacy. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025.

Wanqi Xue, Youzhi Zhang, Shuxin Li, Xinrun Wang, Bo An, and Chai Kiat Yeo. Solving large-scale extensive-form network security games via neural fictitious self-play. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Wanqi Xue, Bo An, and Chai Kiat Yeo. Nsgzero: Efficiently learning non-exploitable policy in large-scale network security games with neural monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 4646–4653, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24611–24624. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

Simon Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Peter Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, et al. A survey on vision-language-action models: An action tokenization perspective. *arXiv preprint arXiv:2507.01925*, 2025.

Zewei Zhou, Tianhui Cai, Yun Zhao, Seth Z.and Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *arXiv preprint arXiv:2506.13757*, 2025.

Zhengyuan Zhou, Ryo Takei, Haomiao Huang, and Claire J Tomlin. A general, open-loop formulation for reach-avoid games. In *2012 IEEE 51st IEEE conference on decision and control (CDC)*, pp. 6501–6506. IEEE, 2012.

Zhengyuan Zhou, Wei Zhang, Jerry Ding, Haomiao Huang, Dušan M Stipanović, and Claire J Tomlin. Cooperative pursuit with voronoi partitions. *Automatica*, 72:64–72, 2016.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

A Detailed Prompt

For agent i, PursuitVLA serializes the query $q_i = (M, L, S_i, C_{1:i-1}, i)$ into a multimodal prompt. The top-down map M depicts the traversable geometry, obstacles, and current agent locations. The task instruction L specifies the controlled role, objective, and game rules, while the ego state S_i records the agent's current position and recent state–action history. For pursuers, $C_{1:i-1}$ contains the waypoint tokens already generated for earlier pursuers in the current joint decision; it is empty for the first pursuer and omitted for the evader.

The output follows a fixed reasoning-and-action format. The reasoning segment covers three elements: a scene description of agent positions and nearby obstacles, an assessment of traversable regions and route geometry, and multi-agent interaction reasoning that predicts relevant teammate and opponent motion. It is followed by six tokens from the 184-action vocabulary, each mapped deterministically to a waypoint displacement. Figure 6 shows the complete role-conditioned prompt structure for both pursuers and the evader at the same decision step.



Figure 6: Prompt templates provided to different agents at the same decision step, from top to bottom: pursuer 1, pursuer 2, and the evader. The ego agent is marked in red and teammates, when present, are marked in blue. Circles represent pursuers, squares represent evaders, and gray regions denote obstacles. Colored lines and boxes are included only for visualization and are not part of the model input.

B Environment Details

All environments are represented as top-down traversability images with continuous agent coordinates. The training set contains 12 maps adapted from Valorant layouts, while the unseen set contains eight maps adapted from CS:GO layouts.

B.1 Training Maps

The 12 training maps are used for action-codebook construction, expert-data generation, supervised fine-tuning, and reinforcement fine-tuning.

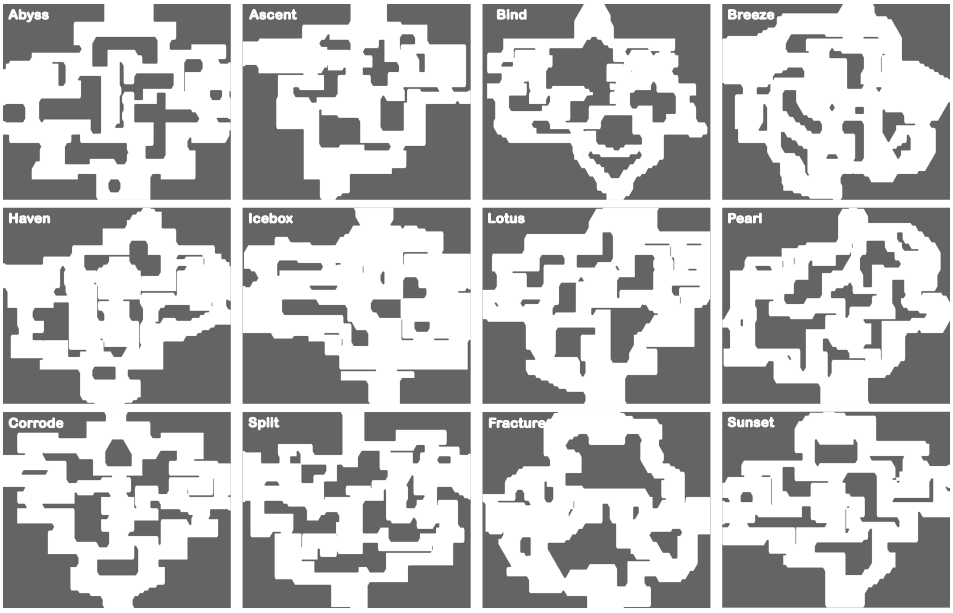


Figure 7: The 12 maps used for expert-data generation and policy training.

B.2 Unseen Maps

The eight unseen maps are excluded from codebook construction, expert generation, SFT, RFT, hyperparameter selection, and checkpoint selection.

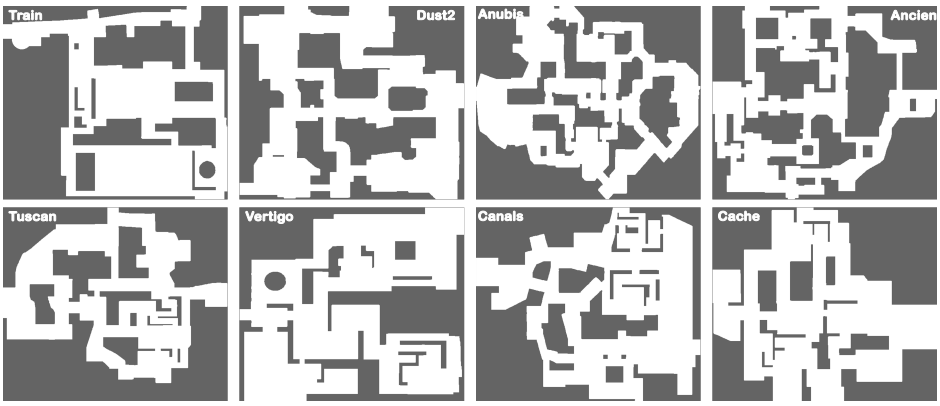


Figure 8: The eight maps reserved for cross-map zero-shot evaluation.

B.3 Locally Perturbed Maps

We manually add or remove local obstacle regions in eight training maps while retaining their broader layouts. Every method uses matched initial conditions before and after each edit. Graph-

dependent baselines retain the graph, distances, and policy inputs derived from the original map, while occupancy-map planners receive the edited map.

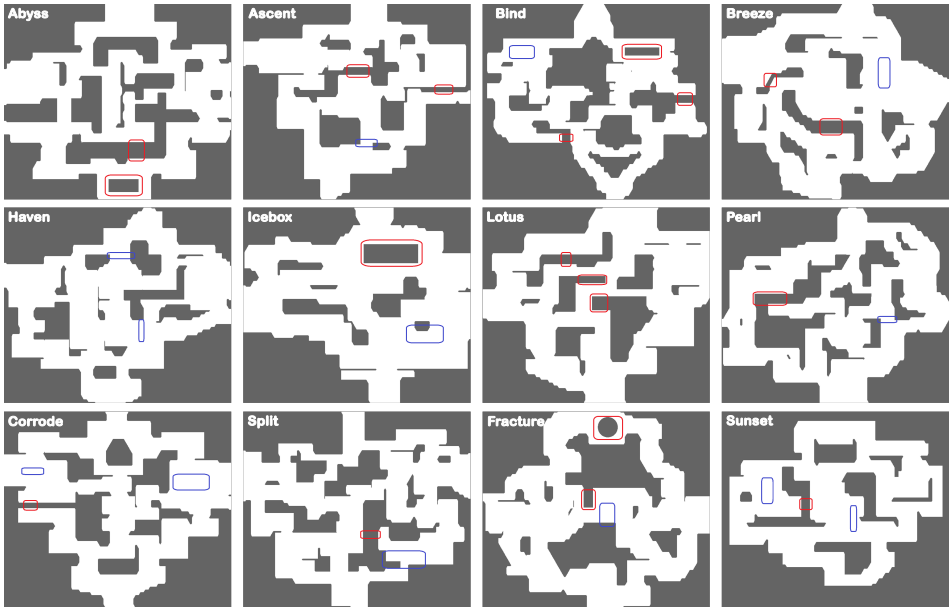


Figure 9: Original and locally perturbed map layouts used for the robustness evaluation. Edited regions are marked with boxes. Red denotes added obstacle regions and blue denotes removed obstacle regions.

B.4 Updated-Representation Evaluation

The main perturbation experiment evaluates adaptation without map-specific representation reconstruction. To separate this effect from policy generalization, we additionally evaluate the frozen graph policies with map representations rebuilt from each perturbed layout. EPG and MAPPO use the rebuilt graph and graph-derived input features without updating their policy parameters, while PursuitVLA receives the updated top-down map without additional training. For this evaluation, all pursuer methods are tested against the same DP evader recomputed from the perturbed map. DP is also recomputed on each perturbed map and is reported separately as a map-specific oracle. All methods use the same perturbed layouts, matched initial conditions, and evaluation seeds as the main perturbation experiment.

Table 2: Zero-shot transfer under updated map representations. EPG and MAPPO receive graph representations rebuilt from each perturbed map, but their learned policy parameters remain frozen. PursuitVLA receives the updated visual map without additional training. All pursuer methods are evaluated against a DP evader recomputed on each perturbed map. The DP pursuer is also recomputed and is shown as a map-specific oracle rather than as a zero-shot policy. Capture rate is reported in percent, and absolute change is reported in percentage points.

Method	Original capture rate ↑	Perturbed capture rate ↑	Absolute change (p.p.)
EPG (rebuilt graph, frozen policy)	69.4	60.6	-8.8
MAPPO (rebuilt graph, frozen policy)	32.2	28.7	-3.5
PursuitVLA (updated visual map)	90.5	85.9	-4.6
DP (recomputed oracle)	100.0	100.0	0.0

C Inference Latency Across Team Sizes

We measure inference latency on a single NVIDIA A100 40 GB GPU with batch size one and the same decoding configuration used in evaluation. Latency is reported per synchronous joint decision step, including autoregressive reasoning and action-token generation for all pursuers in the team.

Table 3: Mean joint-action inference latency across team sizes on a single NVIDIA A100 40 GB GPU with batch size one.

Team size	2v1	4v1	6v1	8v1	10v1
Joint-action latency (s) ↓	0.31	0.56	0.82	1.04	1.30

Because PursuitVLA generates pursuer decisions sequentially, mean latency increases approximately linearly from 0.31 s in 2v1 to 1.30 s in 10v1 as the number of pursuers grows.

D Qualitative Transfer

Across the unseen examples, the pursuers approach through complementary free-space routes and converge around the evader's path. The same policy and action vocabulary produce different waypoint sequences as the obstacle geometry changes. On the matched Bind example, PursuitVLA reroutes after the edit, while DP continues to use the graph and value function computed for the original map. These trajectories illustrate individual behaviors. All quantitative results in the main text use the complete evaluation set.

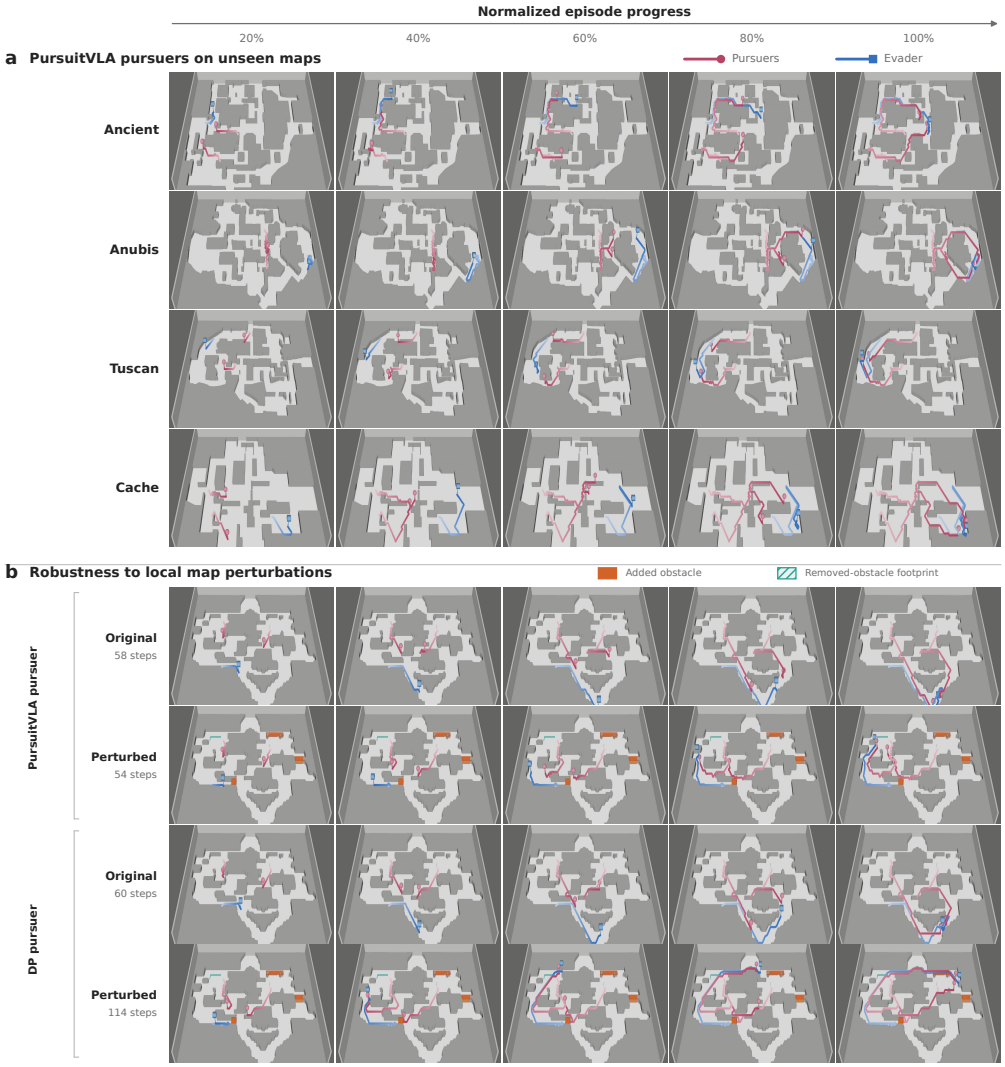


Figure 10: **Trajectories on unseen and perturbed maps. a**, PursuitVLA trajectories on Ancient, Anubis, Tuscan, and Cache. **b**, Matched PursuitVLA and DP trajectories on Bind before and after a local obstacle edit. All four Bind rollouts use seed 6 and identical initial positions. PursuitVLA captures in 58 and 54 steps on the original and edited maps, while DP captures in 60 and 114 steps. Magenta denotes pursuers and blue denotes the evader.

E Additional Qualitative Analyses

E.1 Representative 2v1 Pursuit–Evasion Scenario

Figure 11 illustrates how the shared policy grounds each role in the same scene while producing role-specific decisions. The agents localize teammates and opponents in ego-centric coordinates, identify obstacles that constrain direct motion, and infer likely motion directions from recent trajectories. Pursuer 1 selects a detour around a blocking obstacle and plans a route complementary to its teammate's predicted motion. The evader instead reasons about pressure from both pursuers and selects a feasible escape corridor. The resulting goals are emitted as waypoint tokens and executed by the same downstream controller used throughout the experiments.

E.2 Qualitative Comparison on Training Maps

Figure 12 compares trajectories under identical initial conditions on training maps. PursuitVLA and the map-specific DP policy form complementary approach routes around obstacles, while EPG,

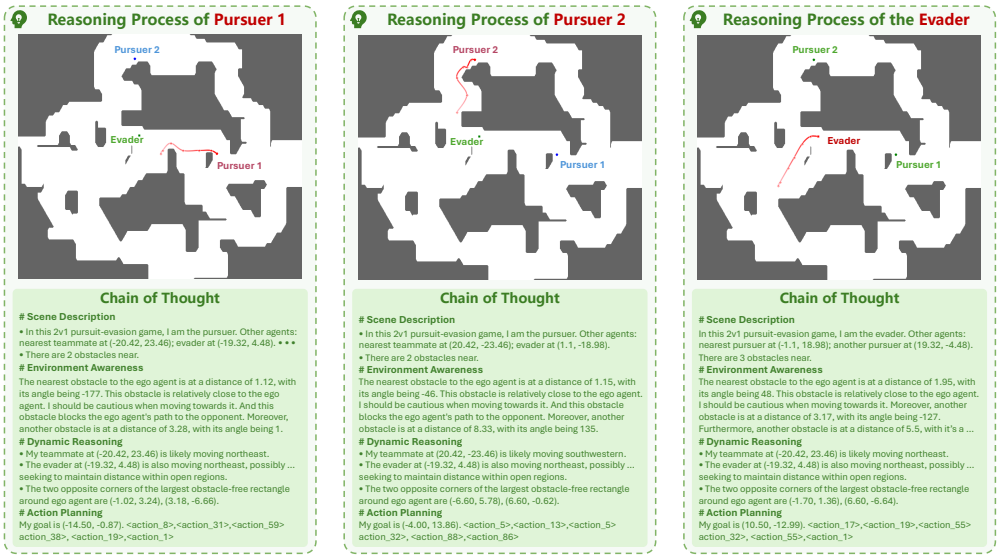


Figure 11: Representative 2v1 pursuit–evasion episode controlled by PursuitVLA. The visualization shows the generated reasoning and planned trajectories for both pursuers and the evader. The ego agent is red and teammates, when present, are blue. Circles represent pursuers, squares represent the evader, and black regions denote obstacles.

Voronoi-guided pursuit, and Shortest Path show less consistent enclosure in this example. The prompted Gemini 3.1 Pro and GPT-5.4 pursuers fail to capture the evader within the 200-step horizon. The figure complements the aggregate training-map results in the main paper and is not used as a separate quantitative evaluation.

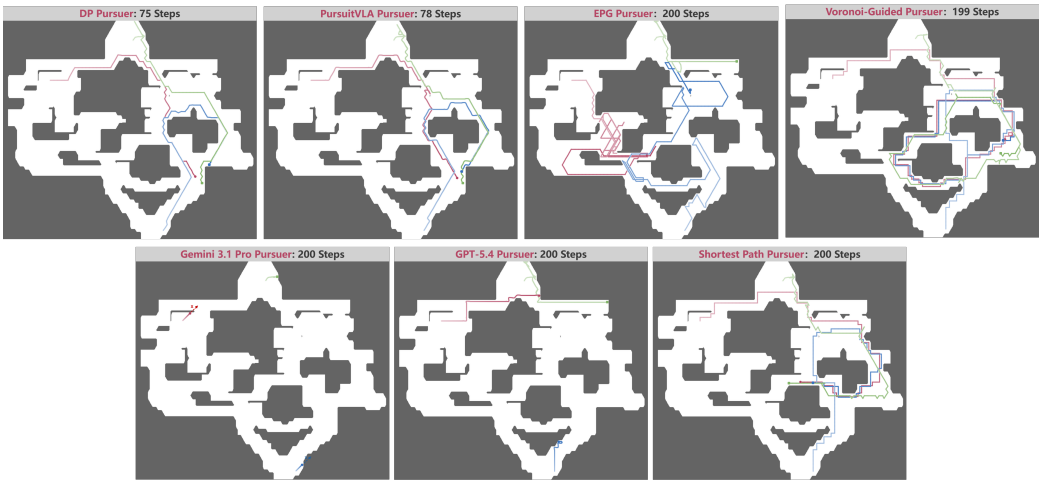


Figure 12: Qualitative comparison with baseline methods on a training map under identical initial conditions. Panel titles report the pursuer policy and episode length; 200 steps indicates that the evader was not captured within the evaluation horizon. Red and blue denote the two pursuers, while green denotes the evader.

E.3 Zero-Shot Generalization Cases

Figure 13 provides additional trajectories on unseen maps whose obstacle geometries do not appear during training. PursuitVLA uses the same checkpoint and action vocabulary in every case.

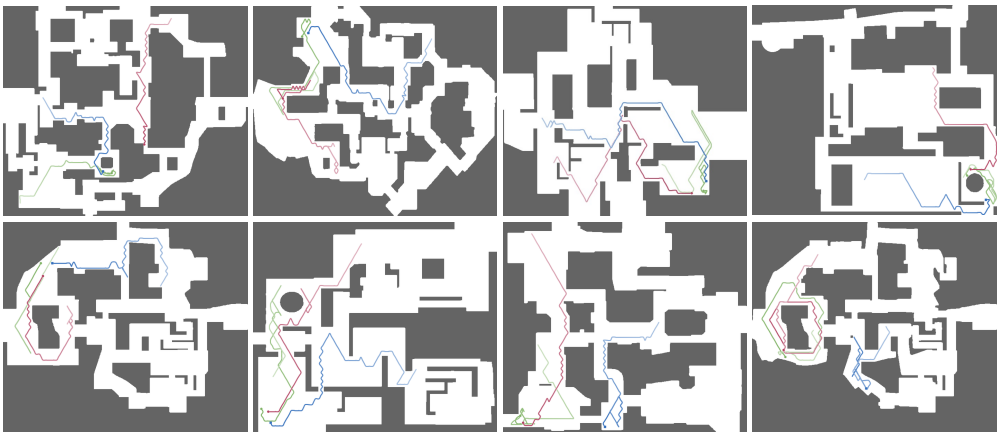


Figure 13: Representative zero-shot pursuit–evasion cases on unseen maps, with PursuitVLA controlling the pursuers. Red and blue denote the two pursuers, while green denotes the evader.

E.4 Robustness on Perturbed Maps

Figure 14 compares PursuitVLA and the DP evader before and after a local obstacle edit. On the edited map, the DP evader continues to act from the graph and value function computed for the original layout and enters the modified dead-end region. PursuitVLA conditions on the edited visual observation and reroutes through the available free space.

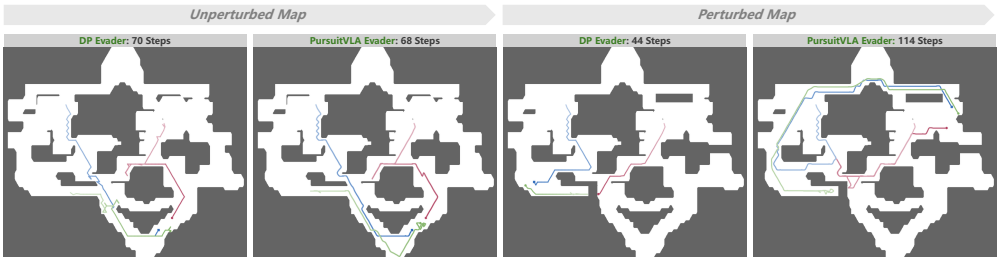


Figure 14: Evader behavior on an original and locally perturbed map against DP pursuers. The left column shows the original layout. The right column shows the edited layout, where the DP evader retains the original graph representation and PursuitVLA responds to the current visual geometry. Red and blue denote the two pursuers, while green denotes the evader.

F Additional Experiments: Head-to-Head Results

We evaluate pairwise pursuer–evader matchups in the 2v1 setting. Figure 15 reports an auxiliary head-to-head matrix collected with Qwen-72B and GPT-5 as the two prompted-model opponents. The main aggregate evaluation in Fig. 3 instead uses the updated Gemini 3.1 Pro and GPT-5.4 baselines. Rows correspond to pursuers and columns correspond to evaders. The heatmap shows how performance changes with opponent strength rather than only against a fixed reference opponent.

PursuitVLA maintains high capture rates across the evaluated evaders and approaches the map-specific DP pursuer in these matchups. The learned graph policies and geometric planners degrade more strongly against the more effective evaders. Qwen-72B and GPT-5 attain lower capture rates as pursuers in this auxiliary pairwise evaluation.

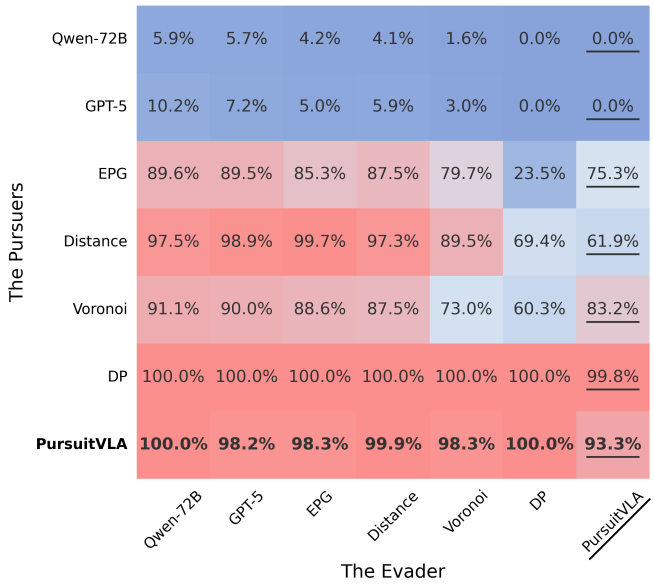


Figure 15: Auxiliary pairwise pursuer–evader matchups in the 2v1 setting. Rows denote the pursuer method and columns denote the evader method. Qwen-72B and GPT-5 are the prompted-model opponents used for this matrix. Each entry reports the pursuer capture rate in percent; higher values indicate stronger pursuit performance.

G Data Construction Details

G.1 Action-Codebook Construction

Figure 16 summarizes the conversion from graph-based expert solutions to continuous waypoint supervision. We enumerate all distinct endpoint displacements reachable by valid trajectories of at most 18 hex-grid steps, producing 1,147 candidate vectors. Candidate frequency is computed from the DP demonstrations on the 12 training maps. Vectors occurring in fewer than 0.1% of sampled trajectory segments are removed, leaving 184 displacement tokens. Each retained vector is assigned one dedicated vocabulary item and a deterministic token-to-vector mapping. The eight unseen maps contribute neither trajectories nor frequencies to this process.

At each training target, six successive displacement tokens encode a short future plan. During simulation and robot deployment, the first token is decoded into the next waypoint and executed before the policy is queried again. The other five tokens are not open-loop controls. They supervise future intent within a receding-horizon policy.

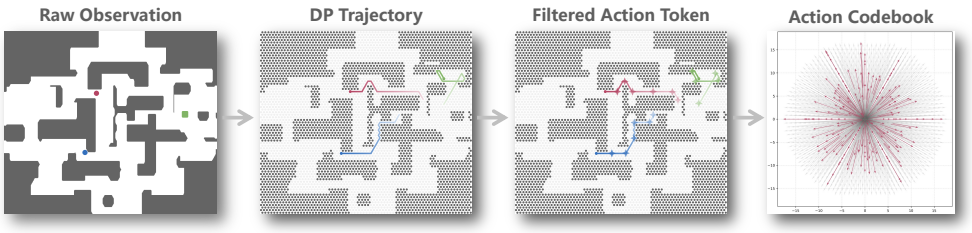


Figure 16: Expert-trajectory conversion, waypoint tokenization, and structured reasoning annotation. DP trajectories on the discretized map are filtered and converted into continuous waypoint displacements. Frequent displacements form the dedicated action vocabulary used for autoregressive prediction.

G.2 Trajectory Conversion and Reasoning Annotation

Each top-down map is first discretized into a hexagonal graph, on which the DP solver produces equilibrium trajectories. After removing oscillatory grid artifacts, we extract key waypoints at turning-angle changes and convert consecutive waypoint displacements into continuous action supervision. A rule-based stage computes relative geometry, obstacle relations, and feasible local routes for every decision state. These values, the filtered trajectory, and the fixed expert displacement are inserted into a structured prompt for Qwen2.5-VL-72B. The annotator explains the supplied action and is not allowed to replace it. This hybrid construction grounds distances, angles, and action targets in simulator values rather than model estimates.

The reasoning schema contains five phases: role and game-context identification, global perception of agents and geometry, obstacle-aware spatial analysis, intent prediction for other agents, and formulation of a waypoint goal consistent with the expert displacement. Role-specific instructions emphasize interception and complementary routing for pursuers and distance preservation and escape-route selection for the evader. Samples with inconsistent geometry, unparseable outputs, or disagreement between the stated goal and the action token are removed.

The resulting 132,895 joint decision states are expanded by role. Each 2v1 state contributes two pursuer-conditioned records and one evader-conditioned record, yielding 398,685 examples. For pursuer records, earlier expert actions populate the autoregressive context.

H Supervised Fine-Tuning Details

We initialize PursuitVLA from Qwen2.5-VL-3B and use a six-token planning horizon. Training proceeds in two stages so that the visual representation is first adapted to the top-down game domain before the language model is optimized for reasoning and action generation. Both stages run on eight NVIDIA A100 40 GB GPUs using DeepSpeed ZeRO-3, BF16, and Flash Attention.

Stage I: visual alignment. The vision tower and multimodal projector are trainable, while the language model remains frozen. This stage aligns visual features with the obstacle layouts, agent markers, and spatial annotations in the training observations.

Stage II: policy fine-tuning. The vision tower is frozen, while the language model and multimodal projector are trained on the role-conditioned reasoning-and-action examples. Pursuer examples use expert teammate prefixes, matching the autoregressive ordering defined in the main paper.

Table 4: Hyperparameters for the two-stage supervised fine-tuning procedure.

Hyperparameter	Stage I	Stage II
Language model	Frozen	Trainable
Vision tower	Trainable	Frozen
Multimodal projector	Trainable	Trainable
Learning rate	1.5×10^{-5}	1.5×10^{-5}
Per-device batch size	4	8
Gradient accumulation	2	1
Warmup ratio	0.03	0.10
Epochs	3	3
Maximum sequence length	2048	2048
Distributed optimization	ZeRO-3	ZeRO-3
Precision	BF16	BF16
Flash Attention	Enabled	Enabled

I Reinforcement Fine-Tuning Details

I.1 Reward Function Design

For pursuer-role responses, the final reward combines the SeqAC task score and a format term:

$$r_i^j = 0.9 r_{\text{task}}^{i,j} + 0.1 r_{\text{format}}^{i,j}. \tag{4}$$

The task score is defined in Eq. 3 using the cached DP remaining-capture value at the intermediate state induced by the policy-generated teammate prefix. A candidate that attains the lowest successor value available from that state receives the maximum task score. Candidates that add predicted capture steps receive graded credit after clipping the successor value at the 200-step episode horizon and normalizing the difference by that horizon.

The binary format term equals one when the response follows the required `<think>` and `<answer>` structure and contains parseable waypoint tokens. Invalid or unparseable responses are executed as no-op actions and receive zero format reward. SeqAC scores only the first waypoint token because only that token is executed before replanning.

RFT without SeqAC. This ablation uses the same SFT initialization, policy-generated teammate prefixes, waypoint action tokens, cached DP value function, format reward, within-group reward normalization, and GRPO configuration as full RFT. The only change is that candidate actions are evaluated from the original global state s_t rather than from the prefix-induced intermediate state $s_t^{(i)}$. Accordingly, the optimal-value reference is

$$\widetilde{V}_i^\star = \min_{a \in \mathcal{A}_i(s_t)} V_{\mathrm{DP}}(p_i(s_t, a)), \tag{5}$$

and the task reward for candidate a_i^j is

$$\widetilde{r}_{\mathrm{task}}^{i,j} = 1 - \frac{\min\left\{V_{\mathrm{DP}}\left(p_i(s_t, a_i^j)\right), H\right\} - \widetilde{V}_i^\star}{H}. \tag{6}$$

The final response reward retains the weighting in Eq. 4.

VLM + MLP policy. The VLM + MLP variant uses the same Qwen2.5-VL-3B initialization, map and state inputs, teammate-conditioned pursuer ordering, expert data, and RFT protocol as PursuitVLA, but replaces autoregressive waypoint-token generation with an MLP head that directly predicts the current pursuer's continuous action. The same waypoint controller and simulator are used for execution; the discrete token-format term is omitted for this continuous-action output.

Evader-role responses use episode survival length as the outcome reward. The DP cache is used only for reward computation on the training maps and is not queried during unseen-map evaluation or deployment.

I.2 Self-Play Data Construction

Before optimization, the frozen SFT policy generates both pursuer- and evader-role trajectories in 600 self-play episodes, comprising 50 episodes on each of the 12 training maps. These trajectories yield 204,660 agent-level queries ordered according to the same autoregressive pursuer sequence used during inference. For each later pursuer, the query contains actions sampled for earlier teammates rather than expert actions.

I.3 Token-Level GRPO Objective

For query q_i, the old policy samples G responses $\boldsymbol{x}_i^j = (x_{i,1}^j, \ldots, x_{i,T_j}^j)$. The response rewards $\boldsymbol{r}_i = \{r_i^1, \ldots, r_i^G\}$ are normalized within the group:

$$A_i^j = \frac{r_i^j - \mathrm{mean}(\boldsymbol{r}_i)}{\mathrm{std}(\boldsymbol{r}_i) + \delta}. \tag{7}$$

For response token $x_{i,t}^j$, the policy ratio is

$$\rho_{i,j,t} = \frac{\pi_\theta(x_{i,t}^j \mid q_i, x_{i,<t}^j)}{\pi_{\theta_{\mathrm{old}}}(x_{i,t}^j \mid q_i, x_{i,<t}^j)}. \tag{8}$$

The token-level objective is

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{j=1}^{G}\frac{1}{T_j}\sum_{t=1}^{T_j}\Bigg(\min\Big[\rho_{i,j,t}A_i^j, \mathrm{clip}(\rho_{i,j,t}, 1-\varepsilon, 1+\varepsilon)A_i^j\Big] - \beta D_{\mathrm{KL}}^{i,j,t}(\pi_\theta \| \pi_{\mathrm{ref}})\Bigg)\Bigg]. \tag{9}$$

The response-level advantage is applied to every generated token. Terminal states generate no further queries.

I.4 Optimization Settings

RFT initializes from the SFT policy and enables end-to-end optimization of the language model, vision tower, and multimodal projector. We use AdamW, BF16, and fully sharded data parallel training on eight NVIDIA A100 40 GB GPUs.

Table 5: Reinforcement fine-tuning hyperparameters.

Hyperparameter	Value
Backbone	Qwen2.5-VL-3B-SFT
Algorithm	GRPO
Optimizer	AdamW
Global batch size	128
Learning rate	1.0×10^{-6}
Weight decay	1.0×10^{-2}
Warmup ratio	0.0
Epochs	15
Maximum sequence length	2048
Group size G	3
KL coefficient β	0.01
Sampling temperature	1.0
Top-p	1.0
Top-k	0
Gradient clipping	1.0
Precision	BF16

J Baseline and Evaluation Details

J.1 Baseline Definitions

The baselines span online geometric planning, graph-based equilibrium methods, multi-agent reinforcement learning, and prompted multimodal models. Their roles and representations are as follows.

- **Shortest Path (Hart et al., 1968).** This online geometric planner selects a shortest collision-free path from each pursuer to the evader's current position on the occupancy map. The resulting path is executed with the same low-level controller used by the other methods.
- **Voronoi-Guided Pursuit (Zhou et al., 2016).** This planner partitions the current free space according to the agents' locations and uses the resulting Voronoi regions to assign complementary pursuit directions. Pursuers move toward their assigned region boundaries, while the evader moves toward a high-clearance location within its region.
- **Dynamic Programming (DP) (Vieira et al., 2008).** DP constructs a discretized graph for each map and solves the corresponding equilibrium value function, producing a map-specific oracle policy under that discretization. It is recomputed for each unseen map, whereas the perturbation evaluation intentionally retains the graph and policy inputs from the original map.
- **Equilibrium Policy Generalization (EPG) (Lu et al., 2025).** EPG is a graph-based reinforcement-learning policy trained across graph instances with DP-derived equilibrium supervision. It receives graph-derived state features and produces graph-level decisions that are converted to continuous motion by the shared execution pipeline.

- **MAPPO (Yu et al., 2022).** MAPPO uses the multi-agent PPO objective with the graph-derived features and sequence representation used by EPG. The pursuers share parameters, while the evader uses a separate policy; reported values average three training seeds.
- **Prompted multimodal models.** Gemini 3.1 Pro (Google DeepMind, 2026) and GPT-5.4 (OpenAI, 2026) receive the same pre-decision map, instruction, and agent-state description as PursuitVLA, but are not fine-tuned on the task. A single model call jointly produces both pursuer actions using a reduced directional action set and parseable displacement blocks; persistent parse failures are executed as no-op actions.

J.2 Executing Graph-Based Baselines in the Continuous Simulator

DP and EPG operate on a discretized graph and output the next graph node, while PursuitVLA predicts a short-horizon waypoint sequence in continuous coordinates. At every environment step, each agent's continuous position is projected onto the graph using the same discretization procedure employed for expert-data construction. The graph policy then selects a target node. Its node center becomes a continuous goal, and A* computes a collision-free path on the occupancy map for the shared low-level motion controller. The graph policy and path are updated at every environment step using the latest agent states. All methods use the same continuous dynamics and execution controller.

DP is reconstructed and solved for each unseen map, where it serves as a map-specific oracle. During perturbation evaluation, DP and EPG intentionally retain the graph, distances, and graph-derived policy inputs from the corresponding original map. This protocol measures whether a fixed graph-dependent policy responds to a local geometric edit without rebuilding its representation.

J.3 MAPPO

MAPPO uses the standard multi-agent PPO objective and the default optimization hyperparameters from DI-engine v0.5.3 (OpenDILab, 2023). It receives the graph-derived features and sequence representation used by EPG. The two pursuers share parameters, while a separate policy is trained for the evader. Each role is trained for 10^6 environment steps against the opposing DP equilibrium policy with three random seeds. The unseen maps are excluded from checkpoint selection, and reported values average the three independent runs.

J.4 Prompted Multimodal Models

Gemini 3.1 Pro and GPT-5.4 receive the same pre-decision map, instruction, and state description as PursuitVLA but are not fine-tuned. One model call jointly generates both pursuer actions. The prompts constrain responses to a reduced directional action set and parseable displacement blocks. Persistent parse failures become no-op actions. This protocol tests executable zero-shot prompting and does not isolate model architecture from action representation.

J.5 Metrics and Matching

Every simulation comparison uses 100 fixed random-seed initial conditions per map. Uncaptured trials receive the 200-step horizon when computing capture or survival steps. Team-size and ablation variants share the same initial conditions, separating policy changes from changes in the state distribution.

K Outdoor Deployment Details

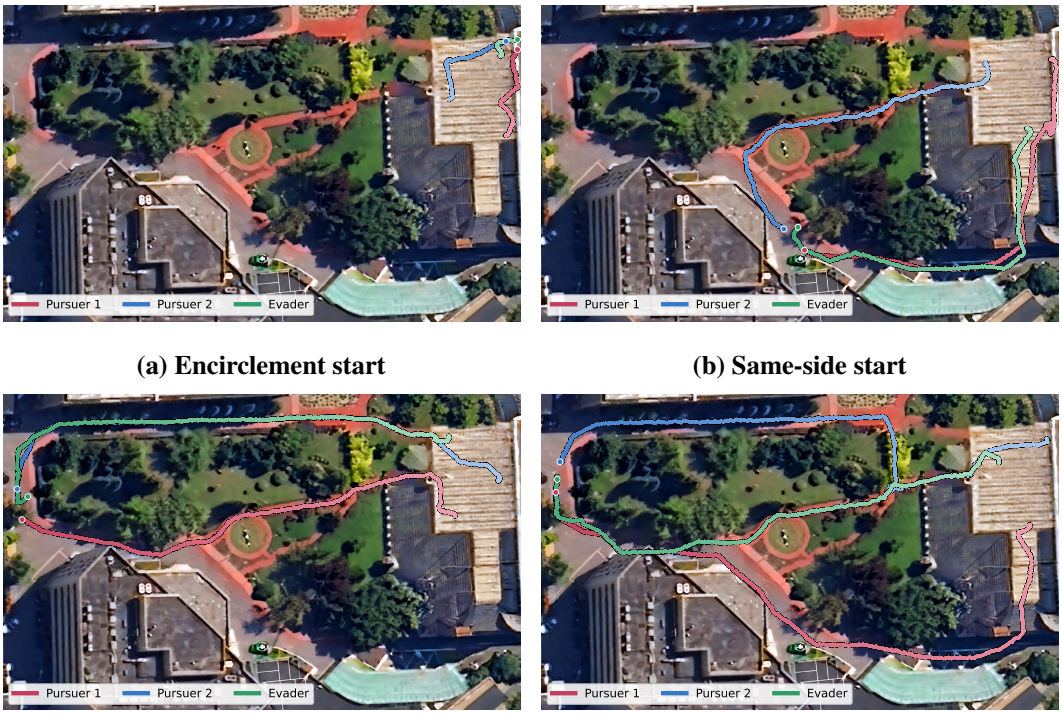
The pre-mapped campus site measures approximately 120 m by 60 m. Point-LIO provides odometry and small-gicp registers LiDAR scans to the map. A local mission server fuses all robot poses, while a remote policy server constructs the top-down observation and queries PursuitVLA at 2 Hz. The first token of each six-token pursuer plan is transformed directly from map coordinates into a global goal pose. The mission server dispatches both pursuer goals synchronously, and a DP policy supplies the evader goal. The robot-side navigation stack uses A* (Hart et al., 1968) or jump point search (Harabor & Grastien, 2011) over occupancy and cost maps and tracks paths using model predictive

path integral control (Williams et al., 2017). Robot-side visual sensing is used only for local obstacle avoidance and is not included in the global observation provided to PursuitVLA.

The four physical configurations are fixed before evaluation. All five trials per configuration contribute to capture rate. Mean capture time is computed over successful trials only. The deployment uses a pre-built map, server-side policy inference, and the platform navigation stack. Online map construction and fully onboard autonomy are outside the evaluated setting.

K.1 Outdoor Robotic Trajectories

Figure 17 shows one representative successful trial for each of the four initial configurations. The panels follow the configuration order in Table 1: encirclement, same-side, narrow-entrance, and opposite-side starts.



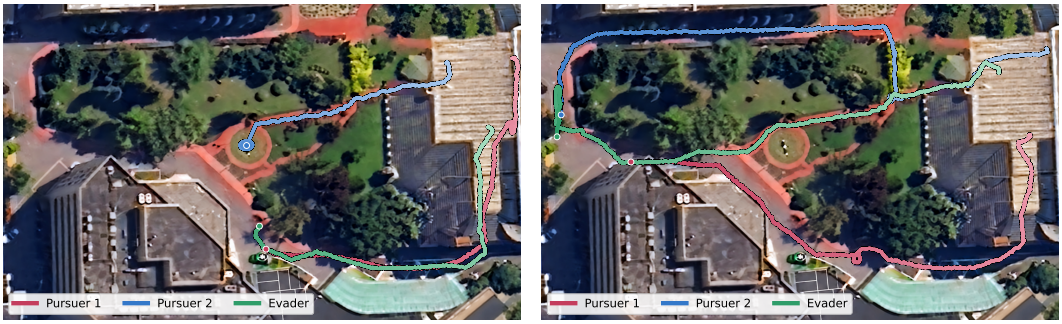
(a) Encirclement start **(b) Same-side start**

(c) Narrow-entrance start **(d) Opposite-side start**

Figure 17: **Representative outdoor pursuit trajectories across the four initial configurations.** Red, blue, and green denote Pursuer 1, Pursuer 2, and the DP-controlled evader, respectively. White-outlined circular markers indicate final positions. Each trajectory is manually aligned to a satellite image of the pre-mapped campus site.

K.2 Outdoor Failure Cases

Figure 18 presents two representative unsuccessful trials. The first trial uses the narrow-entrance start (configuration (c) in Fig. 17): the robot-side obstacle-avoidance system does not reliably detect a low flowerbed, and the pursuer attempts to traverse it and becomes stalled inside the flowerbed. The second trial uses the opposite-side start (configuration (d) in Fig. 17) under a tight time horizon: one pursuer hesitates and loops during its approach, consuming the remaining time before the team can complete the capture.



(a) Low-flowerbed detection failure
Configuration (c): narrow-entrance start

(b) Timeout after pursuer hesitation
Configuration (d): opposite-side start

Figure 18: **Representative failure cases in outdoor robotic pursuit. (a)** From the narrow-entrance start (configuration (c)), a low flowerbed is not reliably recognized by local obstacle avoidance, causing a pursuer to enter the flowerbed and stall. **(b)** From the opposite-side start (configuration (d)), a pursuer hesitates and loops during a time-constrained trial, preventing capture before the episode timeout.

L Exact Numerical Results for Main Figures

Table 6 reports the aggregate values underlying Fig. 3. Table 7 reports the aggregate values underlying Fig. 4. Capture steps assign the 200-step horizon to uncaptured episodes, and all methods use 100 matched initial conditions per map. The prompted multimodal models were evaluated on the unseen maps and therefore do not have training-map reference entries.

Table 6: Comparison of pursuit–evasion methods corresponding to Fig. 3. **(a)**, Pursuer performance on the 12 training maps. **(b)**, Pursuer performance on the eight unseen maps. **(c)**, Evader survival on the eight unseen maps. Capture rate is reported in percent, and step metrics include the 200-step horizon for uncaptured episodes. Bold denotes the best non-oracle result in each column.

(a) Pursuer performance on training maps

	vs. DP evader		vs. PursuitVLA evader	
Agent	Capture rate ↑	Capture steps ↓	Capture rate ↑	Capture steps ↓
Shortest Path (Hart et al., 1968)	30.0	173.5	35.3	168.5
Voronoi-Guided (Zhou et al., 2016)	54.5	152.5	60.8	149.8
MAPPO (Yu et al., 2022)	44.0	148.0	43.0	160.0
EPG (Lu et al., 2025)	75.0	127.0	77.0	129.0
DP (Vieira et al., 2008)	100.0	68.0	100.0	69.5
PursuitVLA (**Ours**)	**90.5**	**110.2**	**91.0**	**99.0**

(b) Pursuer performance on unseen maps

	vs. DP evader		vs. PursuitVLA evader	
Agent	Capture rate ↑	Capture steps ↓	Capture rate ↑	Capture steps ↓
Gemini 3.1 Pro (Google DeepMind, 2026)	5.0	194.8	6.9	195.0
GPT-5.4 (OpenAI, 2026)	8.1	190.6	10.1	190.6
Shortest Path (Hart et al., 1968)	29.0	174.6	34.7	169.4
Voronoi-Guided (Zhou et al., 2016)	53.7	153.7	60.1	150.8
MAPPO (Yu et al., 2022)	32.2	173.4	30.3	177.3
EPG (Lu et al., 2025)	63.0	149.1	59.9	159.3
DP (Vieira et al., 2008)	100.0	69.0	100.0	65.7
PursuitVLA (**Ours**)	**79.6**	**120.5**	**84.3**	**115.2**

(c) Evader survival on unseen maps

	vs. DP pursuer	vs. PursuitVLA pursuer
Agent	Survival steps ↑	Survival steps ↑
Gemini 3.1 Pro (Google DeepMind, 2026)	60.0	60.2
GPT-5.4 (OpenAI, 2026)	60.0	60.2
MAPPO (Yu et al., 2022)	58.9	59.1
EPG (Lu et al., 2025)	56.9	60.4
DP (Vieira et al., 2008)	69.0	120.5
PursuitVLA (**Ours**)	**65.7**	**115.2**

Table 7: Comparison of robustness, team-size transfer, and ablation results corresponding to Fig. 4. **(a)**, Capture rates on the eight training maps used for the perturbation evaluation, before and after local map edits. **(b)**, Capture steps when the 2v1-trained policies are evaluated with 2–10 pursuers. **(c)**, Component ablations evaluated in 2v1. Capture rate is reported in percent, and capture steps include the 200-step horizon for uncaptured episodes. Bold denotes the best result within each directly comparable block.

(a) Robustness to local map perturbations

Agent	Original capture rate ↑	Perturbed capture rate ↑
Shortest Path (Hart et al., 1968)	23.5	24.1
Voronoi-Guided (Zhou et al., 2016)	55.3	54.9
MAPPO (Yu et al., 2022)	32.2	20.1
EPG (Lu et al., 2025)	69.4	45.7
Dynamic Programming (DP) (Vieira et al., 2008)	100.0	81.2
PursuitVLA (**Ours**)	**90.5**	**93.3**

(b) Transfer to larger pursuit teams

Metric	2v1	4v1	6v1	8v1	10v1
SFT capture steps ↓	126.5	122.1	118.0	116.6	115.9
RFT capture steps (**Ours**) ↓	**110.2**	**101.3**	**89.9**	**80.7**	**75.9**

(c) Component ablations

Variant	Capture rate ↑	Capture steps ↓
SFT design		
SFT w/o reasoning	48.3	159.0
SFT one-shot joint generation	60.4	146.3
SFT full	**82.8**	**126.5**
Imitation-learning variants		
Expert-only; expert action	75.1	139.6
All prefixes; fixed expert action	68.3	155.0
All prefixes; conditioned action	**79.8**	**135.1**
RFT design		
RFT w/o SeqAC	80.7	138.9
RFT w/o action tokens	81.9	124.0
VLM + MLP	84.0	122.4
PursuitVLA (**full RFT**)	**90.5**	**110.2**